NASA Contractor Report 179437

Honeywell

IMPLEMENTATION OF A PARALLEL CHANNEL MAXIMUM LIKELIHOOD ESTIMATION ALGORITHM IN A MICROPROCESSOR

By

S.G. Pratt
G.L. Hartmann
D.A. Shaner

Prepared Under
contract NAS4-2578

For:
NASA DRYDEN FLIGHT RESEARCH CENTER

SYSTEMS & RESEARCH CENTER

AEROSPACE & DEFENSE GROUP

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-179437		

4. Title and Subtitle	5. Report Date
IMPLEMENTATION OF A PARALLEL CHANNEL MAXIMUM LIKELIHOOD ESTIMATION ALGORITHM IN A MICROPROCESSOR	December 1979
	6. Performing Organization Code
7. Author(s) S. G. Pratt G. L. Hartmann D. A. Shaner	8. Performing Organization Report No. 79SRC109 /H-1512
9. Performing Organization Name and Address Honeywell Systems and Research Center 2600 Ridgway Parkway Minneapolis, Minnesota 55413	10. Work Unit No.
	11. Contract or Grant No. NAS4-2578
12. Sponsoring Agency Name and Address NASA Dryden Flight Research Center P. O. Box 273 Edwards AFB, California 93523	13. Type of Report and Period Covered Final Report, October 1978 to October 1979
	14. Sponsoring Agency Code

15. Supplementary Notes

Contract Monitor: Mr. Kenneth J. Szalai

16. Abstract

The NASA Dryden Flight Research Center has flight tested a modern adaptive control law in an F-8C aircraft using a remote digital augmentation technique. A ground-based computer containing the adaptive control algorithm estimated aircraft parameters using parallel channel maximum likelihood estimation (PCMLE). Control gains were computed from surface effectiveness estimates and uplinked to the test vehicle. The ground computer was a general-purpose mini-computer with 16-bit word length and floating-point hardware. The successful flight test of this identifier motivated a study to assess the feasibility of implementing the PCMLE algorithm in a microprocessor. The 16-bit TMS-9900 microprocessor of Texas Instruments was selected.

The PCMLE algorithm was programmed using FORTRAN in conjunction with assembly language routines to minimize cycle time and maintain accuracy. The algorithm uses 10K (16-bit word) of memory and has a cycle time of 77 msec. The algorithm has been evaluated in a real-time simulation operating at 10 sps. The report describes the development of the microprocessor software and its simulation performance.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Microprocessor Maximum likelihood estimation F-8 PCMLE algorithm	Distribution unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
UNCLASSIFIED	UNCLASSIFIED	187	

* For sale by the National Technical Information Service, Springfield, Virginia 22151

CONTENTS

CONTENTS (concluded)

LIST OF ILLUSTRATIONS

LIST OF TABLES

IMPLEMENTATION OF A PARALLEL CHANNEL MAXIMUM LIKELIHOOD ESTIMATION ALGORITHM IN A MICROPROCESSOR

Stephen G. Pratt
Gary L. Hartmann
Donald A. Shaner
Honeywell Systems and Research Center

SUMMARY

This report documents the parallel channel maximum likelihood estimation (PCMLE) algorithm implemented on a TI990 microprocessor for the NASA Dryden Flight Research Center under Contract NAS4-2578. This effort used the PCMLE software developed under Contract NAS4-2344 (Reference 1) and tailored it for a microprocessor. The remainder of this report describes the development of the software and includes flow charts and program listings of the final code. For the theory relevant to this software, the reader is referred to NASA CR-2880 (Reference 2).

SECTION 1

THE PCMLE ALGORITHM

Description

The PCMLE algorithm is based on standard maximum likelihood estimation theory as applied to longitudinal short-period F-8C dynamics. Instead of using the usual iterative calculation to maximize likelihood functions, however, it uses the parallel channel implementation shown in Figure 1. Several Kalman filter channels operate at fixed locations in parameter space. Likelihood functions are computed for each. Sensitivity equations are then solved only for the maximum likelihood channel and used to interpolate from there to the final parameter estimate with a single Newton/Raphson parameter correction. This fixed structure avoids real-time iterations and eliminates convergence problems.

Theoretical identifiability results were used to determine the number of parameters that could be identified with small test inputs. This accuracy analysis also provides insight into the number and location of the filter channels.

Nominally, five parallel channels are used to handle the F-8C aircraft over its entire operational flight envelope. The locations of these channels in M_{δ_O}-M_α parameter space are shown in Figure 2. Up to four parameters:

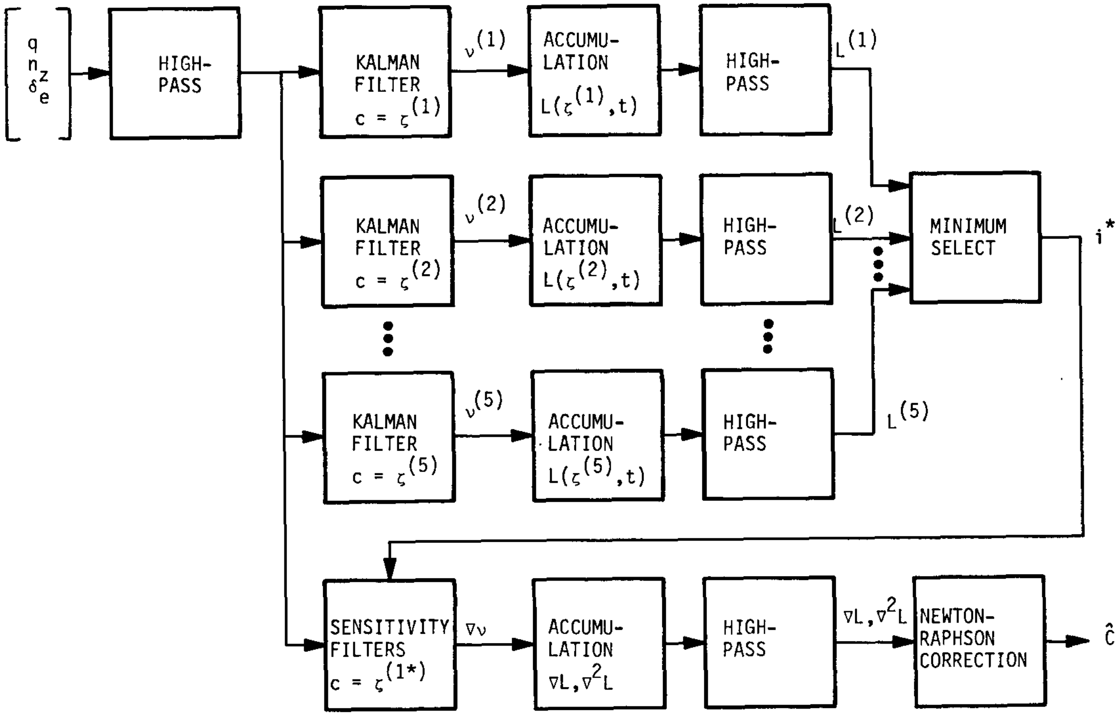


Figure 1. Basic PCMLE Algorithm

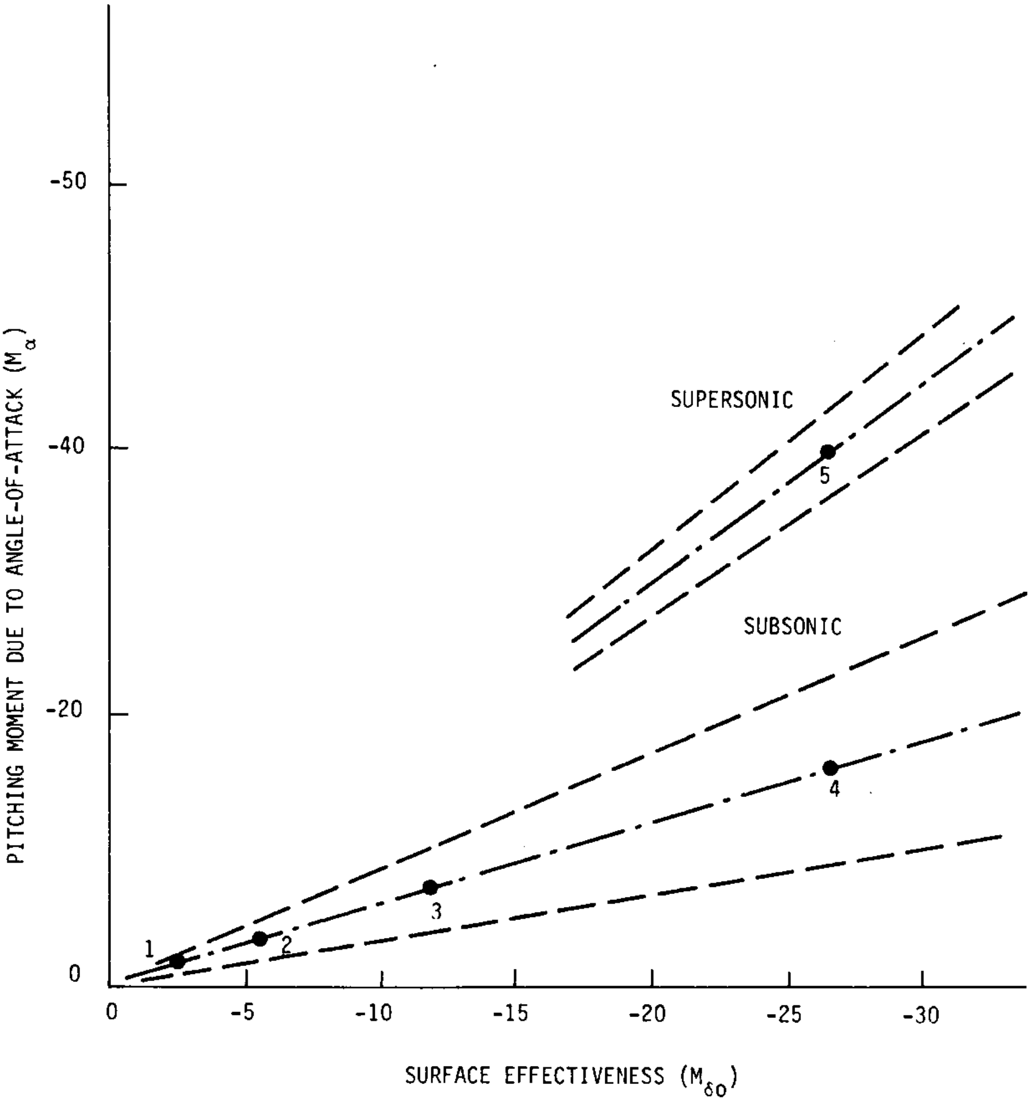


Figure 2. F-8C Identifier Channel Locations

surface effectiveness ($M_{\delta o}$), pitching moment due to angle-of-attack (M_α), airspeed (V), and normal force due to angle-of-attack ($z_\alpha V$) can be estimated. Estimation accuracy depends strongly on the signal levels in the control loop. For the small test signals producing less than 0.05g rms of normal acceleration, errors are 10 to 20 percent in $M_{\delta o}$ and 20 to 30 percent in M_α and V, which are typical in six-degree-of-freedom simulation runs. Theoretical accuracy analyses confirm these error levels.

Software Structure

An overview of the PCMLE software organization is shown in Figure 3. The computations are divided into a background (non-real-time) segment to define and initialize Kalman filter channels and a real-time segment to process sensor data for parameter identification. Calculations performed in each of these segments are divided among a number of subroutines, as shown in Table 1. The functions of each segment and their input/output structures are briefly described below. The core required for PCMLE is 5655 locations for subroutines plus 2730 locations for storage arrays.

Initialization

The initialization of PCMLE is performed in non-real-time with a call to subroutine NRTIC. This subroutine reads the input data deck and user options (UX and LX arrays) and checks the input data reasonableness. It then defines the specified numbers of channel, each at its specified parameter values. Each channel is a four-state Kalman filter. The states are pitch rate, total angle-of-attack, gust angle-of-attack, and elevator surface position. The two measurements are pitch rate and normal acceleration.

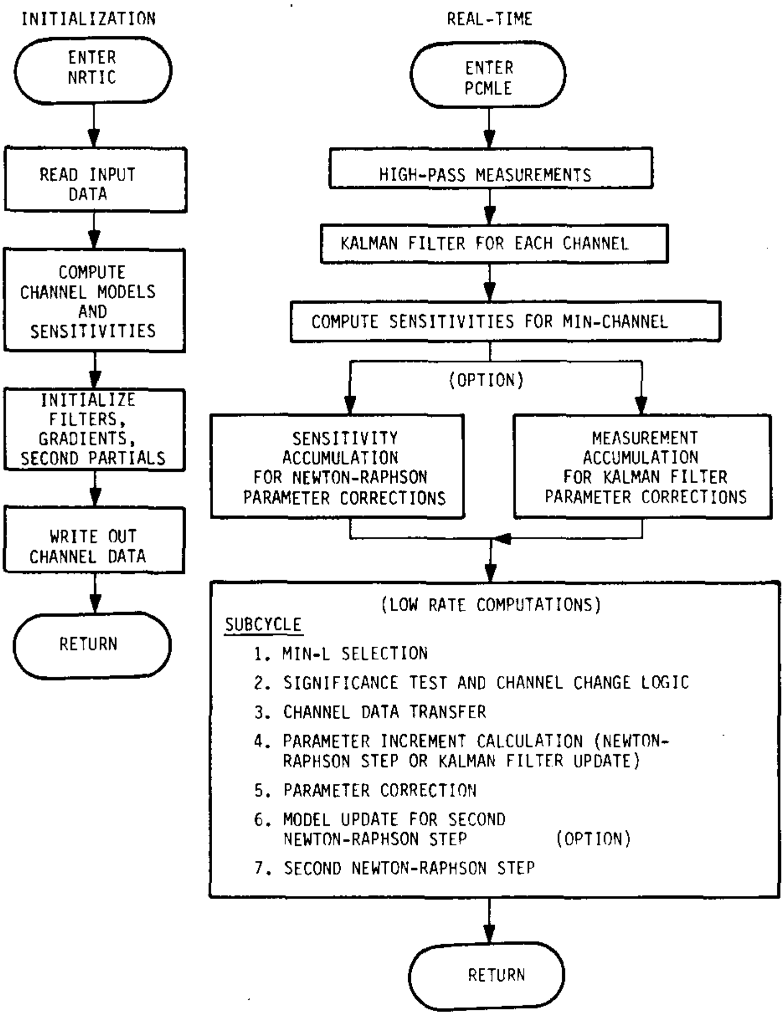


Figure 3. PCMLE Software Structure

TABLE 1. PCMLE SUBROUTINES

Non-Real-Time Subroutines	Functions	Core Required (decimal)
NRTIC	Main executive routine for non-real-time operation. Reads data to define number and location of channels, number of parameters estimated, sample rate, etc. Performs all initialization with calls to other subroutines.	930
MODEL	Defines the system matrices and sensitivities for the discrete four-state model described in Section 1.	638
FHIC	Computes high-pass filter coefficients and initializes filter states for each measurement to be high-passed.	13
DIAK CAL	Solves Ricatti equations for the Kalman filter gains of a discrete system, using double iteration procedures.	401
POLES, QRCALL QR, HESSEN	Computes eigenvalues for channel models and their Kalman filter dynamics.	730
Real-Time Subroutines		
PCMLE	Main executive routine for parallel channel MLE real-time computations.	1767
FH	High-pass filter applied to measurements.	14
TSIG	Produces test signal and two random numbers for simulated sensor noise.	41
FILT	Performs fourth-order Kalman filter update computation.	95

TABLE 1. - Concluded

Real-Time Subroutines	Functions	Core Required (decimal)
SENS	Performs a sensitivity filter update for a given parameter.	176
ACCNR	Accumulates likelihood gradients and approximate second partials for a Newton-Raphson parameter correction.	177
SENS2	Performs sensitivity filter updates for "roving" channel of second Newton-Raphson step.	176
ACCNR2	Accumulates likelihood gradients and approximate second partials for second Newton-Raphson step.	177
ACCK	Accumulates measurements for a Kalman filter parameter correction.	32
KBF	Performs a Kalman filter parameter correction.	288

Real-Time Operation

All real-time computations are executed with a call to PCMLE once per sample time. During each call sampled values of pitch rate, normal acceleration, and elevator servo position are input to the algorithm. The algorithm outputs the selected parameter estimates.

Fixed-Form Definition

A fixed-form version using selected PCMLE options was defined for microprocessor implementation. The fixed form consisted of the nominal five channels and estimated four parameters (M_δ, C_2, C_3, C_4) with a single Newton-Raphson step using the four-state model of the pitch axis. All subroutines and common blocks not used by this version were deleted. In addition, the five remaining subcycles (see Figure 3, Low Rate Computation, Subcycle 1-5), were recoded to be separate subroutines. The initialization software was also configured to be a stand-alone program. The set of program subroutines used by the fixed-form version is given in Table 2, with listings of the PCMLE routines appearing in Appendix A. The next section describes the transformation from the fixed-form version to a TI990-based, real-time system.

TABLE 2. FIXED-FORM PCMLE ROUTINES

Routine	Function
MLEIC	Stand-alone initialization program. Call NRTIC, etc. as listed in Table 1, Non-real-time subroutines. Provide output file of common block data.
F8SIM	Simulates F-8C dynamics with fourth-order linear time-varying model. Test signal and pilot pitch rate input allowed. Stores pitch rate, normal acceleration, and elevator servo position samples on output file.
FFMAIN	Executive program for fixed-form PCMLE. Requires files generated by MLEIC and F8SIM as input. Provides printed output of all major program variables and plots of M_δ estimate, M_α estimate, channel index, and five likelihood functions.
PCMLE	Algorithm executive subroutines
FILT	Kalman filter
SENS	Sensitivity filter
ACUM	∇L and $\nabla^2 L$ accumulation
FH	High-pass filter for input data
CYC1-CYC5	Subcycles 1-5 as listed in Figure 3
Data Files	Contents
DATR	MLEIC Output
F8DATA	F8SIM Output

SECTION 2

REAL-TIME SYSTEM DEVELOPMENT

System Organization

The first step in setting up the real-time system was to determine the appropriate use of the available computers. In this phase, basic processing requirements were examined and assigned to the machine best suited for each function. An important consideration in the division of tasks was data transfer capability: how should information be passed to minimize the need for additional hardware/software development and, in the case of real-time transfer, to minimize time? Four computers were at our disposal. Table 3 shows the relevant attributes of each.

On the basis of the information in Table 3, a block diagram was constructed (see Figure 4). Because it is the most powerful computer, the H6080 was used for as much FORTRAN program development and non-real-time calculation as was convenient. The TI990 AMPL was used as an interpreter between the H6080 and TI990 PS, with cassette as a medium common to all three. It was on this machine that compilations of real-time FORTRAN programs were performed, and formatted data tapes translated to a representation that the TI990 PS could more easily decode. Assembly language routines, developed on either of the TI990 machines, were linked with FORTRAN library routines and the compiled FORTRAN routines to provide a real-time system object tape that was loadable on the TI990 PS. The TI990 PS performed all real-time processing of the PCMLE algorithm

TABLE 3. COMPUTER SYSTEM ATTRIBUTES

Category	H6080	TI 990 AMPL	TI 990 PS	SDS 9300
Program Development: Editor, ease of use	Very good	Good	Fair	
Language support	FORTRAN	FORTRAN Assembly	Assembly	FORTRAN Assembly
Real-Time capability	None	Adequate	Adequate	Good
Program memory	90K	24K	16K	24K
Mass storage	• Disk • Cassette	• Disk • Cassette	Cassette	Magnetic tape
User I/O	80-character Printer/cassette terminal (ASR) via modem	72-character VDT ASR Front panel	ASR Front panel	• Card reader • Front panel • Sense switch
Machine interconnection			SDS 9300	TI990 PS
Accessibility	Time-shared	Shared	Dedicated	Dedicated

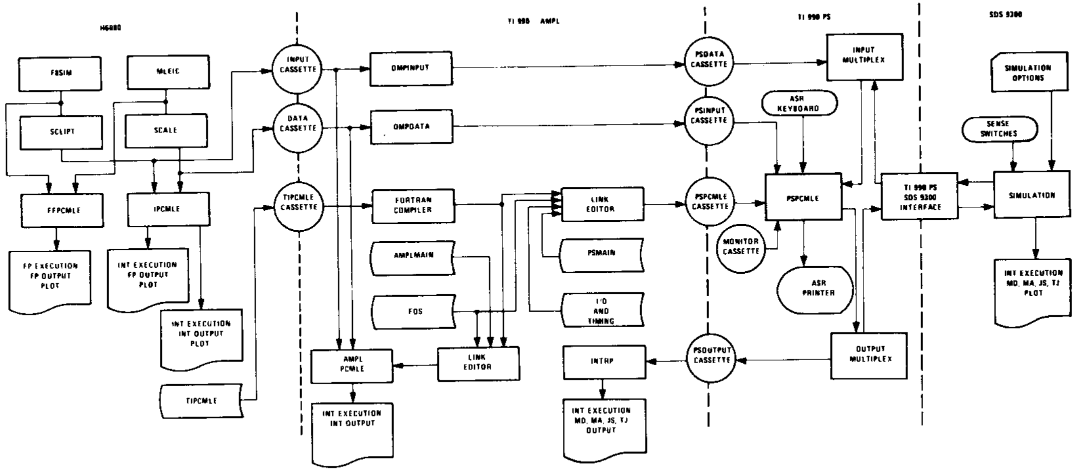


Figure 4. Task Definition and System Interconnection Block Diagram

and was connected via hardware/software interface to the SDS 9300, which was responsible for real-time simulation and translation of TI990 PS output into formatted hard copy output. A summary of the required software, listed by machine, is presented in Table 4.

Constraints

The next step was to investigate how the fixed-form program should be tailored to run on the TI990. The three major constraints were memory, execution speed, and programming simplicity. The TI990 PS has 16K words of memory, 4K of which are required by the resident monitor. The TI990 AMPL system, on which the development work was done, has 24K words with approximately 8K required for its more powerful operating system.

The fixed-form program was compiled on the AMPL system and linked with the appropriate FORTRAN run time routines, which provided floating-point arithmetic and input-output (I/O) capabilities among others. It was found that the program could be made to run in the available 12K, but only if the general I/O functions were abandoned; software modules were created for our specific needs.

Attaining an adequate execution speed, however, was a more difficult problem. Appendix B summarizes the investigation of recoding options for the FORTRAN program, showing the tradeoffs among the various arithmetic modes. As a result of this study the integer mode was selected as the most viable alternative. Although the per-cycle time of 77 ms (average) falls short of the

TABLE 4. INTEGER PCMLE ROUTINES

Machine	Routine	Function	Data Files	Contents
H6080	SCALE	Converts fixed-form initialization data to be compatible with integer PCMLE. Uses MLEIC output file as input. Scales constants, derives new constants and scale factors. Write output file.	DATA (Disk → Cassette)	Scaled contents and scale factors
	SCLIPT	Scales F8SIM floating-point output into integers for use by integer PCMLE.	INPUT (Disk → Cassette)	Scaled simulation output
	IMAIN	Executive program for integer PCMLE on H6080. Requires files generated by SCALE and SCLIPT as input. Provides printed output of all major program variables in either integer or rescaled real represention for comparison with fixed-form. Provides the same plots as FFMAIN.		
	PCMLE, FILT, SENS, ACCUM, FH, CYC1-CYC5	Same function as in fixed-form (Table 2). Subroutines modified for emulation of TI990 integer operation.	TIPCMLE (Disk → Cassette)	TI990 PCMLE subroutines as edited on H6080
	MD, MS, S, SD	H6080 FORTRAN routines written as FOS emulators.		

TABLE 4. - Continued

Machine	Routine	Function	Data Files	Contents
TI990 AMPL	DMPDATA	Reads cassette tape containing formatted output from SCALE into integer PCMLE common blocks. Dumps common blocks in ASCII-coded hexadecimal in 80-character records (with a record count for each block) onto cassette.	DATA (Cassette = Diskette) PSDATA (Cassette)	From H6080 Initialization data for TI990 PS
	DMPINPUT	Reads cassette tape containing formatted output from SCLIPT into input array Y. After each record (set of three samples) is read, it is dumped in ASCII-coded hexadecomal onto one cassette record.	INPUT (Cassette = Diskette) PSINPUT (Cassette)	From H6080 Simulated input data for TI990 PS
	INTPR	Reads cassette tape containing ASCII-coded hexadecimal output from PCMLE algorithm and prints formatted data on line printer.	PSOUTPUT (Cassette)	From TI990 PS
	AMPLMAIN	Executive program for integer PCMLE on TI990 PS. Requires cassette generated by DMPDATA for initialization. See Figure 5 for I/O options and program flow chart.	CLIME (Cassette)	Linked object tape of PSMAIN, PCMLE subroutines, FOS, and I/O and Timing subroutines
	TIPCMLE (Cassette⇒Disk)	PCMLE subroutines from H6080.		
	MD, MS, S, SD	FOS		

TABLE 4. - Continued

Machine	Routine	Function	Data Files	Contents
TI990 AMPL	INPCAS	Contains routines to read sensor data cassette (if this option is selected) and data initialization cassette.		
	IOASR	Contains routines to read from and write to the ASR terminal printer.		
	CLOCK5	Provides timing control of real-time operation.		
	IO9300	Supports input and output data transfers between the TI990 PS and the SDS9300.		
	OUTCAS	Writes output variables to cassette (if this option is selected).		
TI990 PS	PSPCMLE (Cassette)	From TI990 AMPL		
	MONITOR (Cassette)	System executive program		
			PSDATA (Cassette)	From TI990 AMPL
			PSINPUT (Cassette)	From TI990 AMPL
			PSOUTPUT (Cassette)	Stand-alone PCMLE output

TABLE 4. - Concluded

Machine	Routine	Function	Data Files	Contents
SDS 9300	SIM9300	Real-time simulation program. Can be run in stand-alone test mode or with any combination of input to and output from the TI990 PS. Execution is alterable via sense switches while program is in progress. Based on program F8SIM.		
	TI990GET	Supports transfer of data buffer from TI990 PS to SDS 9300.		
	TI990PUT	Supports transfer of data buffer from SDS 9300 to TI990 PS.		
	CONVTI	Converts a TI990 PS format floating-point number received over the interface to its SDS 9300 equivalent.		

target sample rate (50 ms; 20 sps), it allows comfortable operation at 10 sps, which should be an adequate rate for identification purposes. Assembly language programming of the algorithm is always an option if a higher rate is required, but it forces the loss of programming simplicity. The advantage of maintaining a basic FORTRAN coding should not be surrendered lightly. For a program of this complexity, assembly coding would render the program unreadable, difficult to modify, and tedious to debug. For these reasons the FORTRAN integer coding was chosen for implementation.

Software Programming

The real-time system software development began with four related programming efforts that progressed in parallel. They were:

1. Modification of fixed-form program to allow for integer computation in subroutines FILT, SENS, ACUM, and elsewhere as demanded for consistency.

2. Scaling of initial conditions and derivation of scale factors for use in fundamental operation subprograms.

3. Development of the FORTRAN-callable fundamental operation subprograms (FOS) in TI990 assembly language.

4. Development of the FORTRAN-callable I/O and timing subprograms to be used for communication with and control of the algorithm in both test and real-time modes.

Modifications of Fixed-Form Program

Timing of the fixed-form program on the TI990 AMPL recealed that four-parameter estimation was not a realistic goal, even with significant reduction in "fundamental operation" time. This observation dictated the first modification to the program: to rewrite ACUM, CYC3, CYC4, and CYC5 for two parameters only. This eliminated unneeded gradient accumulations and allowed for the replacement of a costly 4 x 4 matrix inversion with a more modest 2 x 2 inversion.

The next step was to restructure the two-parameter version in order to effectuate the use of integer arithmetic. The predominate equation form is that of the inner product (FILT, SENS, ACUM, FH). A typical inner product would appear in the program like this:

```
              .
              .
X(K, 7)  =  D(K, 9)*X(K, 3) + F(K, 8)*X(K, 9) + F(K, 12)*X(K, 10)
              .
              .
```

The first attempt at restructuring the program was to convert such equations from floating-point to extended-integer arithmetic. To avoid overflow upon multiplication, information in all operands must be limited to a single word. In order to force extended multiplication, at least one operand must be extended. The convention of single-word variables and extended constants (with a zero upper word) was adopted. The information in the operands was scaled, so it was necessary to adjust the products (via multiplication/division by a scale factor) prior to summation. Finally, the upper word of the extended summation (most significant part) was assigned to the single-word, lefthand variable. The resulting equation would look like this:

```
INTEGER X(5, 10), TEMP (2)
INTEGER*4 D(5, 16), F(5, 13), TEMPA
EQUIVALENCE (TEMPA, TEMP)
                 .
                 .
                 .
TEMPA   =  D(K, 9)*X(K, 3)*2 + F(K, 8)*X(K, 9) + F(K, 12)*X(K, 10)/2
X(K, 7) =  TEMP(1)
                 .
                 .
                 .
```

Here the equivalence statement enables access to the first word of the extended summation, TEMPA. All the inner products in the three subroutines previously mentioned were rewritten in this manner. Timing of this "extended integer" version (see Appendix B) revealed an only slightly-improved execution speed and indicated that more comprehensive measures were in order.

From the results of the first restructuring experiment, it was obvious that the second restructuring should include some special-purpose assembly language routines. It was also clear that more floating-point operations would have to be converted to integer. After consideration of the TI990 capabilities and brief analysis of execution speed versus accuracy tradeoffs, the following form was adopted:

```
INTEGER X(5, 10), D(5, 16), F(5, 13), S, Q(88)
 .
 .
 .
X(K, 7) = S(D(K, 9), X(K, 3), Q(18)) +
S(F(K, 8), X(K, 9), Q(19)) + S(F(K, 12), X(K, 10), Q(20))
 .
 .
 .
```

where S(.) is one of the four FOS. The costliest drawback of this form is that products are truncated before accumulation. In the case of inner products of a variable (X) and constant (D, F), this effect is noticeable, but not restrictive in any example that we have tested. There are cases, however, in which cross-products of two variables are needed. In these situations a double-word accumulation is desirable. Function MS (extended integer) was created for this purpose.

Conversion of all high-pass accumulation to integer arithmetic was the next logical step. Contained in subroutines FILT and ACUM, these equations (and associated variables) had maximum magnitudes that could be easily predicted on the basis of those already assumed for the state and sensitivity variables. A typical example of this form is:

```
GL(1) = E1P*GL(1) + GE(1)
```

Because it is an accumulator, the lefthand variable can grow quite large so, in order to preserve precision when it is not, it is necessary to use extended-integer arithmetic. The integerized equation would look like this:

```
INTEGER            E1P, Q(88)
INTEGER*4 GL(2) + GE(2), MD, TSD
          .
          .
GL(1) = MD(E1P,GL1) + SD(GE1, Q(75))
          .
          .
          .
```

After all the equations of this form were converted, the low rate operations (CYC1 - CYC5) were modified to eliminate mixed-mode arithmetic wherever possible. With these changes, the type assignment (i.e., INTEGER, INTEGER*4, REAL) of all program variables and constants was complete.

In order to attain real-time capability it was important that all operations be performed as efficiently as possible. All "on-line" computation should involve at least one program variable; any expression involving only constants should be reduced to a single constant. For this reason, some of the fixed-form expressions were reduced as follows:

ZP(JS, 1)**2 ⇒ ZPS(JS) (CYC2)
S1 = 1. + ZP(JS, 1)/ZP(JSTEMP, 1) ⇒ SZP(JS, JSTEMP) (CYC3)
S1*S1 ⇒ SZP2(JS, JSTEMP) (CYC3)

These arrays could now be pre-calculated as part of the initialization procedure, thus eliminating costly real-mode computation in real time.

With the functional modifications complete, the issue of initialization could be addressed. In order to facilitate the transfer of program constants, it was necessary to redimension all variables and constants to the minimum size required by the two-parameter version. The unnecessarily large general-purpose arrays UX and LX were dropped in favor of the more meaningful variables for measurements (Y) and logical options (CHC, EST, MLE), respectively. The original eight common blocks (VARDAT, DAT, SENSP, MEAS, PARM, TSW, PLIM, FHDAT), which were sorted by use, were resorted by type (LOGL, INT2, INT4, REAL). New variables and constants required as a result of the restructuring were added to the appropriately typed block.

The final modification was to improve program readability. Conditional branches were simplified. "DO" loop limits were adjusted to agree with assigned dimensions. Statement labels were ordered with regular intervals.

Comment card format and equation spacing were standardized. It was in this form that the program was transferred to the TI990 AMPL for debugging. The only changes made from this point were those required for correct communication between the PCMLE program and TI990 assembly language subroutines. The final version of the program can be seen in FORTRAN (Appendix C) and TI990 Assembly (Appendix D).

Scaling

The purpose of scaling is to convert numbers in floating-point representation to an integer representation that retains as much information as possible. In an ideal situation this would imply a "1" in the most significant bit of the binary representation. In the case of constants this ideal can be attained. As an example, let us consider the scaling of the constant:

$$DT_{FP} = 0.1$$

This first step is to determine the "scale factor" or minimum power of two which is greater than the constant. In this case DTSCL = -3 (since $2^{-3} = 0.125 \geq 0.0625 = 2^{-4}$).

Now the appropriate multiplier must be determined. In the TI990 an integer is 16 bits long with one bit reserved for sign, giving it a maximum value of $2^{15} - 1$ and thus a scale factor of 15. In order to take advantage of the precision available, the constant is scaled by the difference of these scale factors. That is,

$$DT_I = 2^{(15-DTSCL)} * DT_{FP}$$

$$= 262144 * 0.1$$

$$= 26214$$

Here the result has been truncated as it would be in its new integer form. In some cases a value will not be a constant but instead will be selected from a set of constants on the basis of some index (channel or parameter). For purposes of efficiency and simplicity it is desirable to choose one scale factor for the set and to scale all constants in the set on the basis of the one scale factor. To do this, the scale factor selected must be the maximum encountered in the given set.

Variables, by their very nature unknown quantities, cannot be scaled to maintain maximum information. They must be scaled small enough to avoid clipping but large enough to contain sufficient information at reduced signal levels.

Fortunately, all the variables involved either have physical limits imposed or are closely linked via equations to the physical variables. Thus by making assumptions for scale factors of a limited subset of the variables (see Table 5), scale factors for the others (TJ, TL, GE, GL, GSQE, GSQL, XY) can be derived.

After scale factors for all constants and variables have been assigned, the scale factors used by the fundamental operation subprograms can be computed. In general, it is desired to assign the product of two numbers (or sum of products) to a third number.

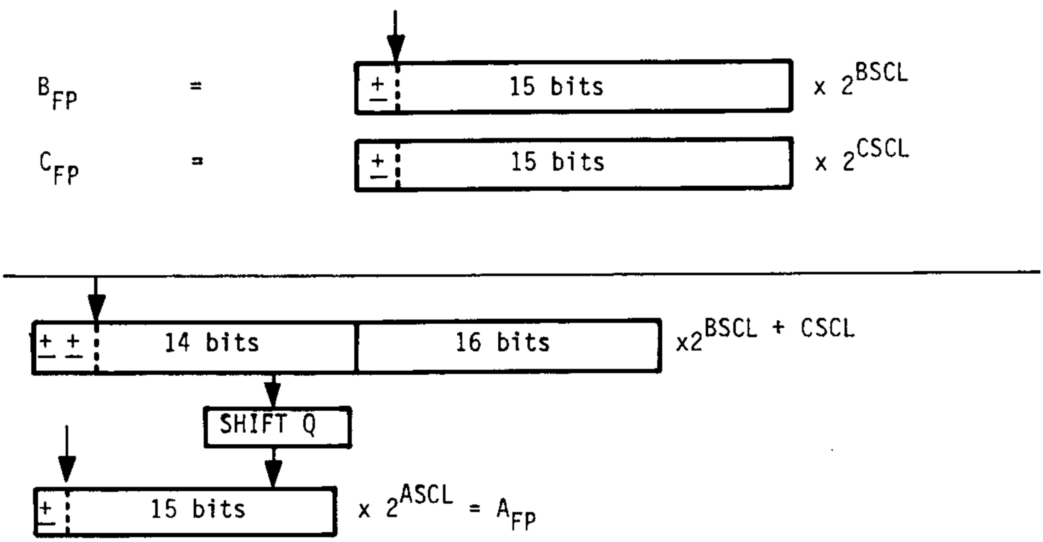
TABLE 5. SCALE FACTOR ASSIGNMENT

X(IC, I) State IC = 1, 5 for channel I = 1, 10 for variable	XS(IP, I) Sensitivity IP = 1, 2 for parameter I = 1, 10 for variable	YP(J) Input J = 1, 3 for variable	
		Scale Factor	
Index (I)	State/Sensitive Variable	X	XS
1.	q_k — Current state/sensitivity at time t_k	1	0
2.	α_{t_k} — Current state/sensitivity at time t_k	-1	-3
3.	α_{g_k} — Current state/sensitivity at time t_k	-2	-4
4.	δ_k — Current state/sensitivity at time t_k	-1	-3
5.	q_{k+1} — Predicted state/sensitivity at time t_{k+1}	1	0
6.	$\alpha_{t_{k+1}}$ — Predicted state/sensitivity at time t_{k+1}	-1	-3
7.	$\alpha_{g_{k+1}}$ — Predicted state/sensitivity at time t_{k+1}	-2	-4
8.	δ_{k+1} — Predicted state/sensitivity at time t_{k+1}	-1	-3
9.	ν_q — current filter residual	1	0
10.	ν_{nz} — current filter residual	8	7
Index (J)	Input Variable		YP
1	q		1
2	n_z		8
3	δ_e		-1

For example:

$$A = S(B, C, Q) + \ldots$$

By the convention previously adopted, it is assumed that A, B, C, have their radix points directly behind their sign bits, with associated scale factors ASCL, BSCL, and CSCL.



The function S is designed to return the first word of the integer product sign $(B*C)* | B | * | C |$ after shifting it by Q bits. Note in the above diagram that the double-word product can contain only 30 bits of magnitude information since B and C both contain 15 such bits. The product has effectively two sign bits, so it must be shifted left one bit in order to position the radix point. In addition, it must be shifted by -ASCL + BSCL + CSCL in order to establish the proper exponent for assignment (or accumulation) to the variable A. The desired Q is therefore given by:

$$Q = 1 - ASCL + BSCL + CSCL$$

With minor variations, most of the Q factors are computed in this manner.

Program SCALE (see Appendix E for program listing and output file) was written to perform the constant scaling and Q factor assignment as described in the preceding paragraphs. In addition to these primary functions, the program calculates the additional constants (SGANS, ANSI, ZPS, SZP, SZPZ) that are required by modified PCMLE program. It receives as input an unformatted BCD file containing the original common blocks created by the initial conditions program. As output it provides a formatted ASCII file containing all program constants, initial conditions, and Q factors. This file is later dumped to cassette for transfer to the TI990 AMPL system.

Fundamental Operation Subprograms

Specifications for the FOS were a side effect of the modification procedure. As a group, the FOS enable integer-mode arithmetic. They perform functions in assembly language that would be either impossible or time-consuming in FORTRAN. Since they are called repeatedly by the PCMLE program, it is essential that they have short execution times. For this reason, it is necessary to leave out the standard TI990 FORTRAN subroutine interface (F$RGMY) and replace it with specific code to compute arguments and return addresses. Without F$RGMY, the programmer must insure that a given argument is passed with a consistent addressing mode (direct or indirect). Through experimentation it was determined that arithmetic expressions and unsubscripted variables were passed by FORTRAN compiler as direct arguments and subscripted variables not identically the first element were passed as indirect arguments.

Examples:

```
DATA J/1/
EQUIVALENCE (NU11, XS(1,9))
X9  =  X(K,9)
```

Q(1)	direct	Q(84)	indirect
SGANS	direct	SGANS(J)	indirect
GE(2)	indirect	GE2	direct
XS(1, 9)	indirect	NU11	direct
X(K, 9)	indirect	X9	direct

In order to minimize the number of required modifications the assumed addressing mode was chosen to agree with the majority of subroutine calls in the program as written (see Appendix B, Table B-2). By studying the examples above, it is apparent that the addressing mode can be controlled without significantly reducing speed or readability . In calls that did not match the assumed mode, the main program was changed to achieve consistency. Although the desired results were attained with TI990 assembly programming, speed was hampered by the lack of some rather simple instructions. One such instruction is a signed two-word product of two signed one-word operands. The current multiply instruction returns a two-word product, but it is only correct if the operands are both positive, or if the result does not overflow the low-order word. For this reason, it was necessary to convert both operands to positive integers, multiply, and assign the appropriate sign to the product. This causes the multiply time to be more than doubled. Another useful but missing instruction is the double-word shift; i.e., the capability to shift bits out the end of one word into an adjacent word. Both of these instructions could be implemented in hardware and are general enough in potential application to justify inclusion in the standard instruction set.

Assembly language listings of the FOS are presented in Appendix F.

I/O and Timing Subprograms

As stated in the paragraph on constraints, the primary effect of limited memory was the loss of standard general I/O capability. As a result, it was necessary to construct special-purpose software to perform the following functions:

- Read from keyboard
- Write to printer
- Read from cassette
- Write to cassette
- Read from interface
- Write to interface

Fortunately, the monitor extended operations (XOP) were available to the user. Used with the proper options and combined with appropriate assembly language, the desired I/O operations were obtainable.

Another requirement for real-time operation was a set of subroutines to synchronize PCMLE with the TI990 line frequency clock. The basic approach was to execute the algorithm while the system automatically counts clock pulses (8.33 ms). After completion of all calculations, program control is passed to one of the timing subroutines, which waits until the appropriate clock pulse (for 100 ms operation, the subroutine waits for the 12th pulse). If the desired cycle time is exceeded, the overflow count is indexed and the subroutine returns control immediately. At termination, maximum time and number of overflows are communicated to the operator.

Assembly language listings with detailed explanations of the I/O and timing routine are presented in Appendix G.

Software Integration

Once programming of the PCMLE algorithm and support software was completed, organization into the various processing packages could proceed. Two basic categories were considered: Off-line software validation modules and the real-time execution module.

Off-Line Software Validation

Because the fixed-form program had gone through many modifications, it was important that the new integer version be checked for correctness. Also, in the likely event that the integer version did not work on its initial trial, adequate data for debugging should be available. For these reasons, the following modules were configured:

1 1. FFPCMLE--Uses FFMAIN (see Table 2) as the executive routine. The output is considered to be the "true" algorithm output. Runs on H6080.

2. IPCMLE--Uses IMAIN (see Table 4) as the executive routine. The output is considered to be an approximate emulation of the TI990 integer version output. Program options provide for direct-integer time histories (for comparison with AMPLPCMLE) or rescaled floating-point time histories (for comparison with FFPCMLE). Runs on H6080.

3. AMPLPCMLE--Uses AMPLMAIN (see Table 4) as the executive routine. The output is identical to that of the real-time processor (PCPCMLE), if it possessed the required I/O routines. Runs on TI990 AMPL.

These three modules provided adequate information to resolve all inconsistencies observed between the fixed-form and integer implementations.

Real-Time Execution Module

In contrast to the off-line processor that provided comprehensive output with little attention to program flow and control, the real-time processor needed to provide minimal output with precise control of operation. Taking these constraints into account, the flow diagram in Figure 5 was constructed. Following this diagram, a main program (PSMAIN) was written to perform calls to the appropriate subroutines in their proper sequence. As the final step PSMAIN was link-edited to the TI990 AMPL with the FOS, I/O, timing, TIPCMLE, and required FORTRAN run time library routines to produce the PSPCMLE object cassette. (See Appendix H for PSMAIN FORTRAN, Appendix I for assembly listing, and Appendix J for link editor listing). This cassette could then be loaded via monitor into the TI990 PS and run in conjunction with the SDS 9300 real-time simulation.

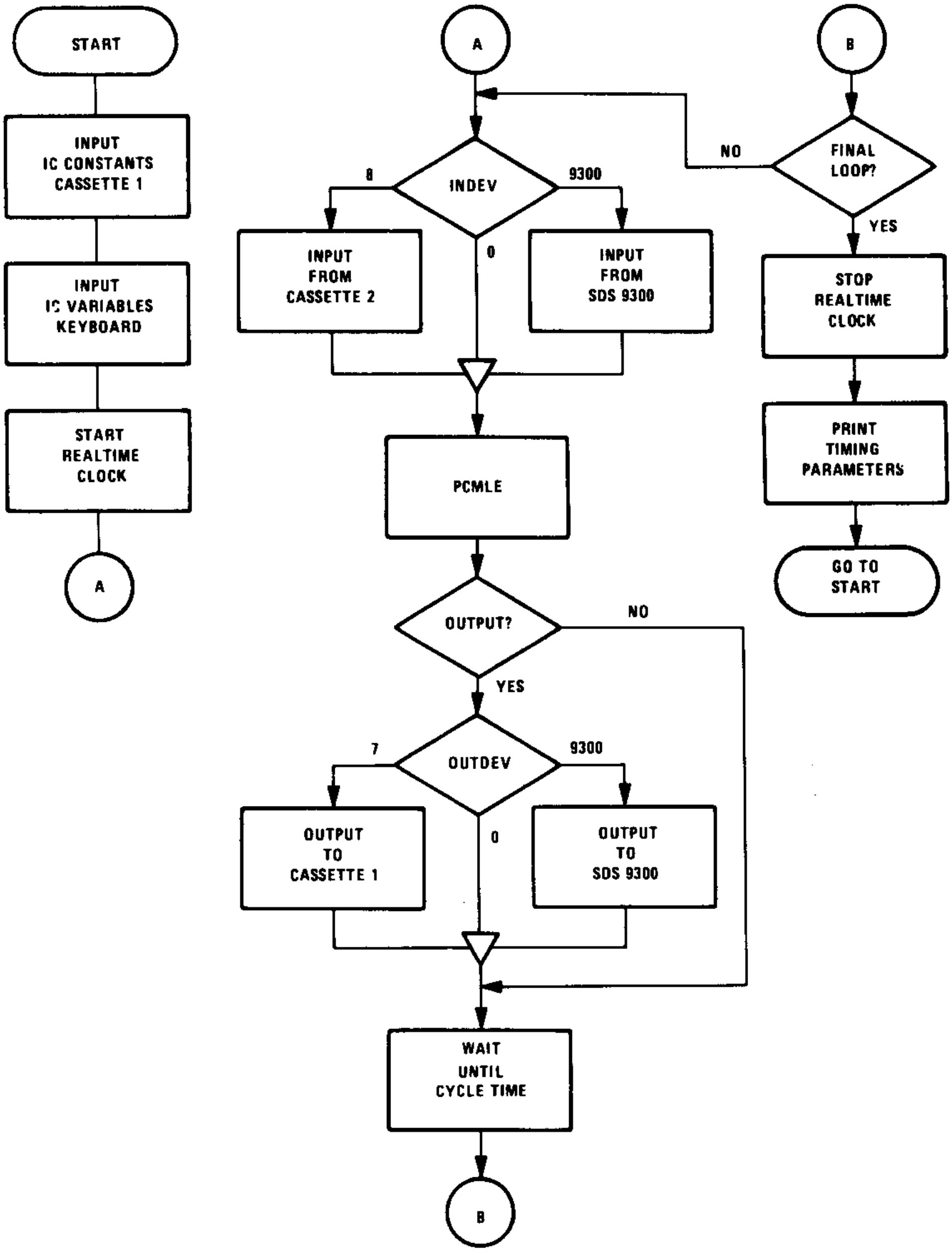


Figure 5. TI990 PSPCMLE Flow Diagram

SECTION 3

ALGORITHM PERFORMANCE

PSPCMLE was required to meet two primary performance objectives: Channel switching times and parameter estimates should be reasonably close to the values output by FFPCMLE. Also, in order to be considered as a real-time processor, it must complete its operation (including I/O) in a time not exceeding 100 ms per cycle. The program was set up to allow independent testing of these requirements. To test accuracy, PSPCMLE was run in stand-alone (cassette I/O) mode and the output data (MD, MA, JS, TJ), after interpretation, was compared with FFPCMLE output. To test cycle time, PSPCMLE was run in a "no I/O" mode, which gave a lower bound (off by the time required for TI990/SDS 9300 transfers) of 83 ms (eight clock pulses). (See Figure 6.)

As the first real-time simulation test case, we used the same demanding run that had already passed the basic accuracy test: acceleration through all five channels with a filtered noise test signal input to elevator position of σ = 0.1 rad. (See Figure 7 for FFPCMLE plotter output and Figure 8 for PSPCMLE plotter output.) In comparing the plots, it can be seen that channel switching is very good with the exception of an early change from four to five. A possible solution to this minor variance is readjustment of the likelihood threshhold for the integer version. Parameter estimates for the integer run are somewhat irregular, an effect that can probably be traced to roundoff error propagation. Over the entire run, however, the estimation error appears to be within a tolerable limit.

.LP••100 .IM•100•102 0100=3EC0 0106 .MP PC=0100 0106 WP=0000 3EC0 .EX	PROGRAM LOADING STAND-ALONE (CASSETTE-TO-CASSETTE)	
IC CS1 =	Ready initialization tape and return	INPUTS
NPARAM = 2	Number of parameters	
NCHAN = 5	Number of channels	
CHC = 1	Change channels?	
EST = 1	Estimate parameters?	
MLE = 1	Perform maximum likelihood estimation?	
NWORDS = 16	Number of 16-bit words to output	
NLOOP = 601	Number of sample loops	
CYCTIM = 100	Desired cycle time	
INDEV = 8 OUTDEV = 7	Input device (8,9300--none) Output device (7,9300--none)	
MAXTIM = 02375 NERROR = 00601	Maximum cycle time Number of cycle time overflows	OUTPUTS
	TIMING (NO I/O)	
IC CS1 =		
NPARAM = 2		
NCHAN = 5		
CHC = 1		
EST = 1		
MLE = 1		
NWORDS =	N/A	
NLOOP = 601		
CYCTIM = 100		
INDEV = OUTDEV =	No inputs or output	
MAXTIM = 00083 NERROR = 00000	Maximum cycle time for algorithm without I/O	

Figure 6. ASR Terminal Input Format with Examples

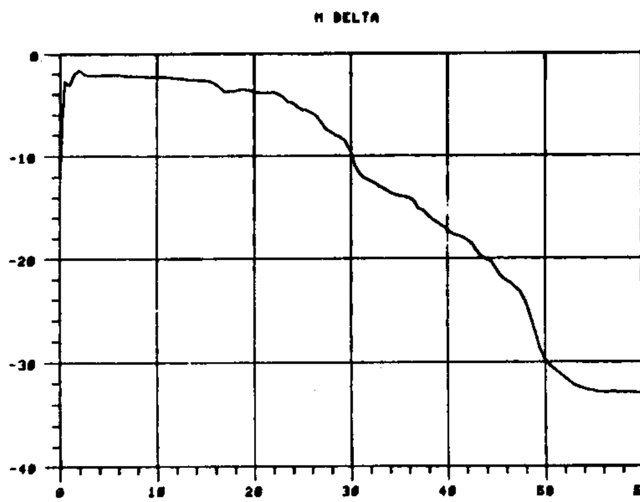


Figure 7a. FFPCMLE: Acceleration (σ = 0.1 rad)

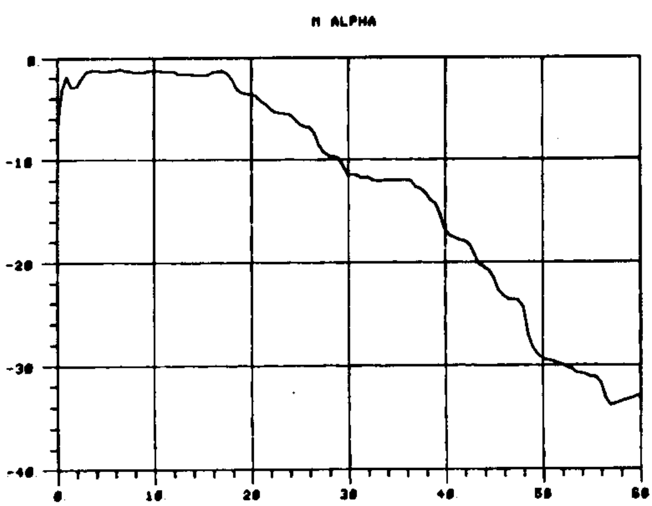


Figure 7b. FFPCMLE: Acceleration (σ = 0.1 rad)

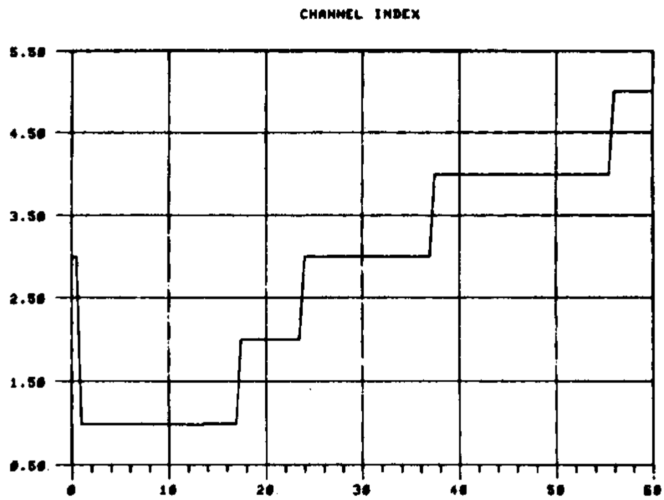


Figure 7c. FFPCMLE: Acceleration (σ = 0.1 rad)

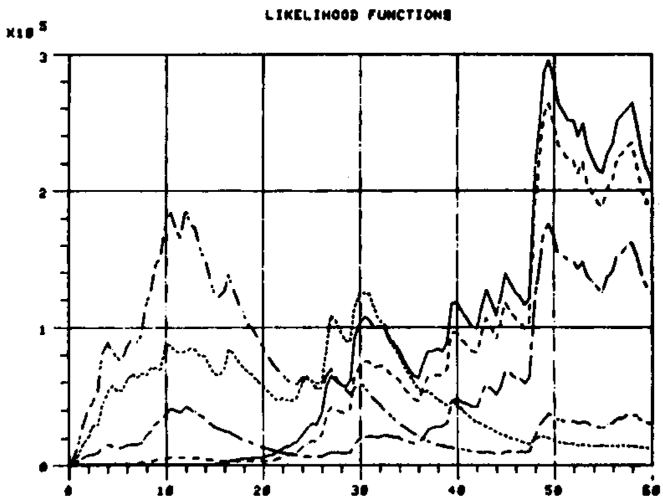


Figure 7d. FFPCMLE: Acceleration (σ = 0.1 rad)

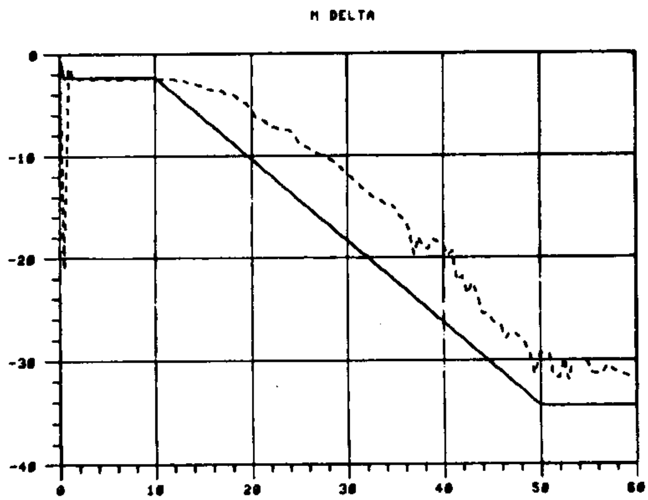


Figure 8a. PSPCMLE: Acceleration (σ = 0.1 rad)

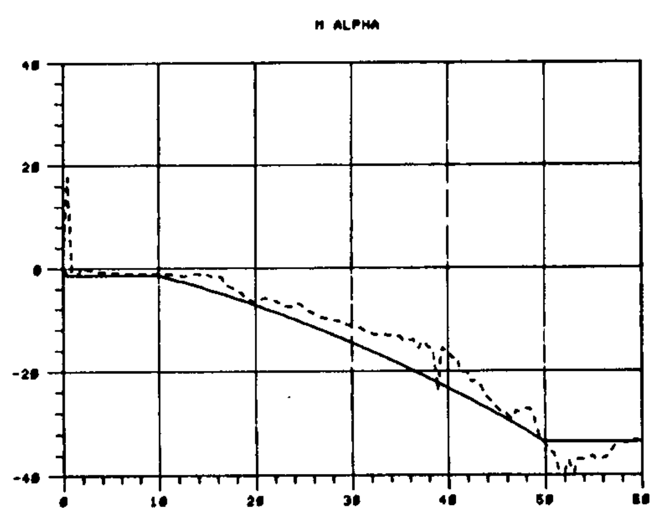


Figure 8b. PSPCMLE: Acceleration (σ = 0.1 rad)

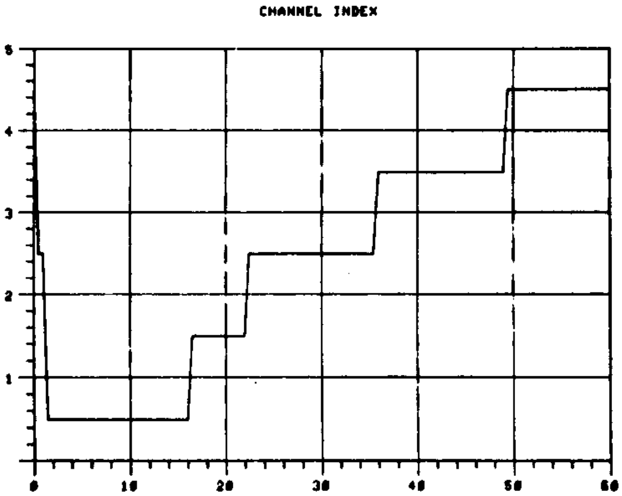


Figure 8c. PSPCMLE: Acceleration (σ = 0.1 rad)

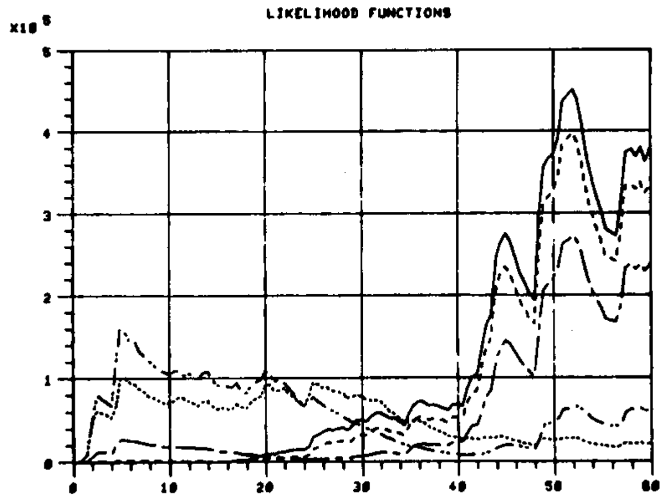


Figure 8d. PSPCMLE: Acceleration (σ = 0.1 rad)

The second simulation test was also an acceleration, but with a test signal of only σ = 0.02 rad. With the reduced input level, roundoff error would be expected to be more significant than in the previous case, and it is. Channel switching in the first five seconds is hampered considerably. In addition, the irregularity increased. Simulation tasks with still lower test signal inputs showed a total lack of reliable estimation capability. (See Figure 9 for plotter output.)

The final simulation test was an acceleration-deceleration with the original test signal level. Observing the minimums of the "actual" and "estimate" plots of M_δ and the effective lag of the estimation process appears to be approximately 5 seconds. This compares favorably with the lag exhibited by the fixed-form version, and should be adequate for the changes in flight condition that the processor will encounter. (See Figure 10 for plotter output.)

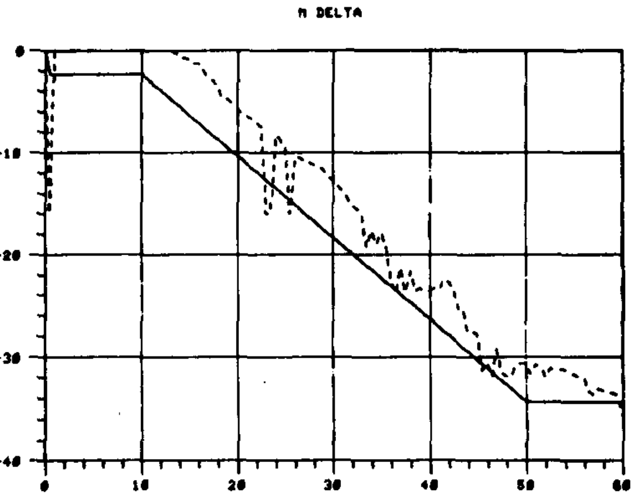


Figure 9a. PSPCMLE: Acceleration (σ = 0.02 rad)

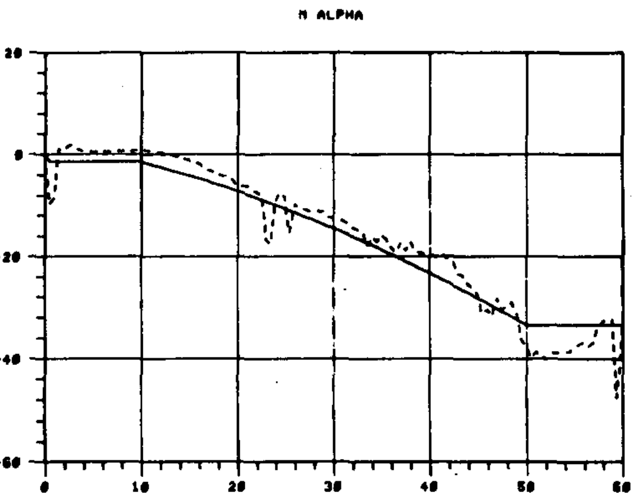


Figure 9b. PSPCMLE: Acceleration (σ = 0.02 rad)

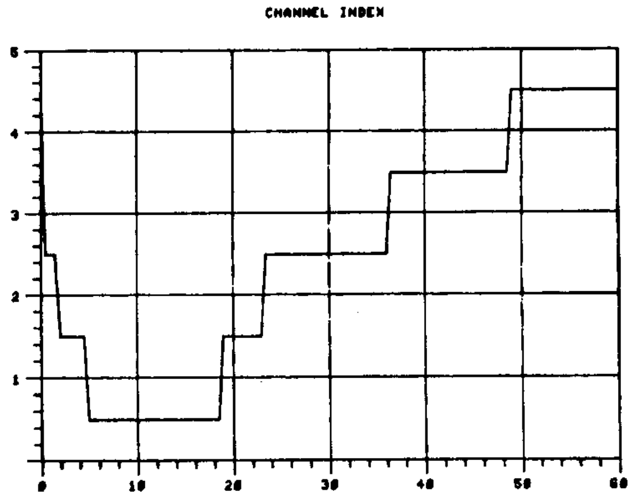


Figure 9c. PSPCMLE: Acceleration (σ = 0.02 rad)

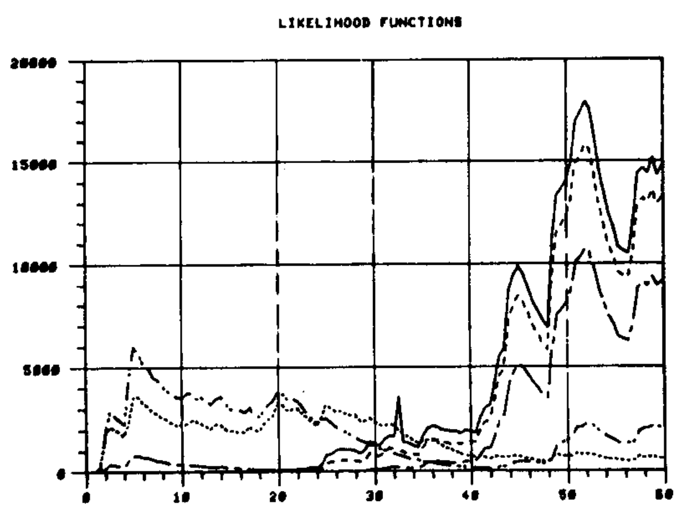


Figure 9d. PSPCMLE: Acceleration (σ = 0.02 rad)

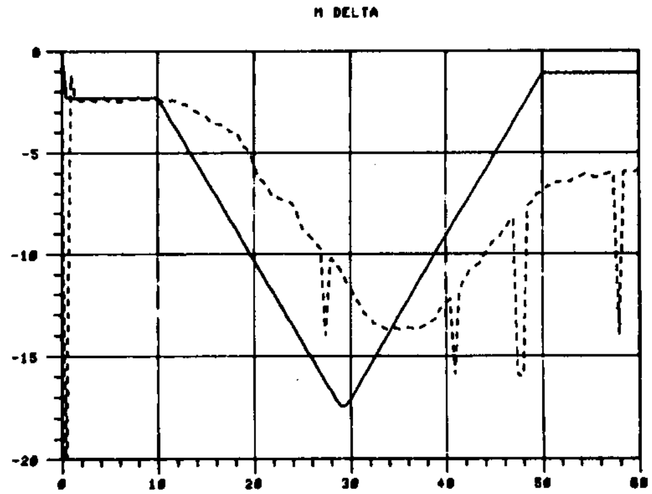


Figure 10a. PSPCMLE: Acceleration/Deceleration (σ = 0.1 rad)

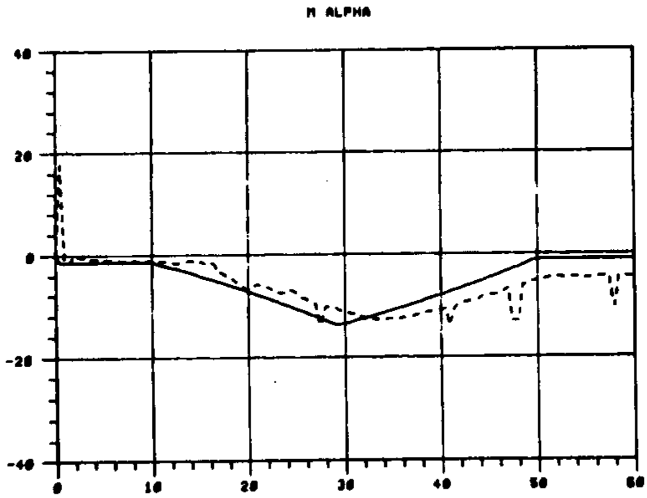


Figure 10b. PSPCMLE: Acceleration/Deceleration (σ = 0.1 rad)

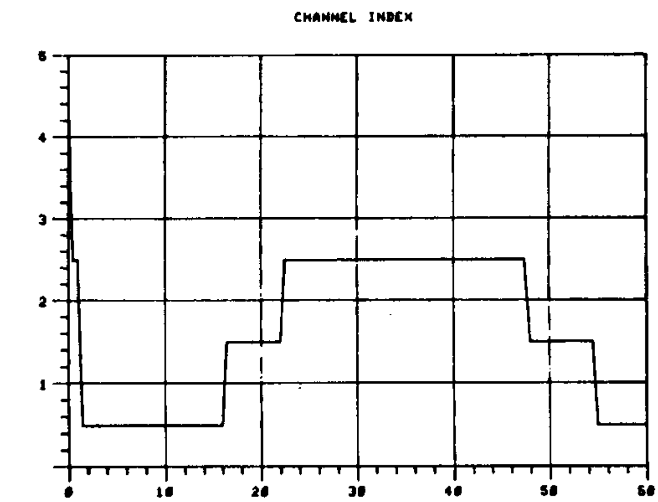


Figure 10c. PSPCMLE: Acceleration/Deceleration (σ = 0.1 rad)

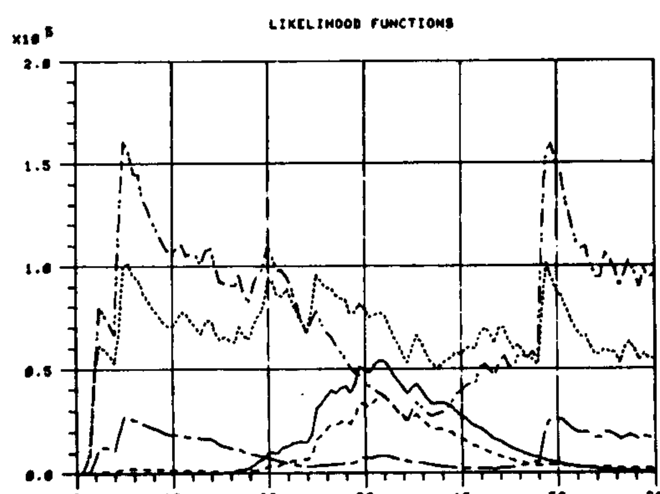


Figure 10d. PSPCMLE: Acceleration/Deceleration (σ = 0.1 rad)

SECTION 4

SUMMARY AND CONCLUSIONS

This report has demonstrated the capability of a microprocessor (TI990) to perform parallel channel maximum likelihood estimation (PCMLE) in real-time. Limitations imposed by the microprocessor execution times and instruction set selection were discussed. A processor conversion methodology that overcame these limitations has been described in detail, with primary attention to the issue of software modification. The implementation has been shown, in simulation testing, to perform in a manner that closely corresponds to the fixed-form version of the algorithm. As microprocessor technology continues to develop, higher sample rates and more complex algorithm implementation with minimal software development will be made possible.

REFERENCES

1. Hartmann, G. L., "PCMLE Software Documentation," Honeywell Report FO459SD, NASA Contract NAS4-2344, Honeywell Systems and Research Center, Minneapolis, Minnesota, October 1976.

2. Hartmann, G. L., et al., "F-8C Adaptive Flight Control Laws," Final Report NASA CR 2880, Honeywell Systems and Research Center, Minneapolis, Minnesota, June 1976.

APPENDIX A

FIXED-FORM FORTRAN LISTING

```
1642T 01   07-16-79   14.245                                                   LABEL PCMLE   PAGE    1
              SUBROUTINE PCMLE                                                   00000070
      C                                                                          00000080
              COMMON/VARDAT/UX(50),LX(50)                                        00000090
              COMMON/MEAS/Y(3),YP(3),XY(2,3),DT,TIME,MODE                        00000100
              COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS   00000110
              COMMON/PARM/ZP(5,4),DZP(4),ZP1,ZP2,ZP3,ZP4,ZIMIN                   00000120
             1   ,NP,NC,JSTEMP,JS,CHCHAN                                         00000130
              LOGICAL LX,NOCHC,NOEST,MLE                                         00000140
              INTEGER CHCHAN                                                     00000150
      C                                                                          00000160
              NOCHC=LX(1)                                                        00000170
              NOEST=LX(2)                                                        00000180
              MLE=LX(9)                                                          00000190
      C                                                                          00000200
      C       REAL TIME INPUTS                                                   00000210
      C                                                                          00000220
              Y(1)=UX(11)                                                        00000230
              Y(2)=UX(12)                                                        00000240
              Y(3)=UX(13)                                                        00000250
      C                                                                          00000260
      C                                                                          00000270
      C       HIGH PASS INPUTS                                                   00000280
              CALL FH(1)                                                         00000290
              CALL FH(2)                                                         00000300
              CALL FH(3)                                                         00000310
      C                                                                          00000320
      C       PARALLEL CHANNELS                                                  00000330
              IF(.NOT.MLE) RETURN                                                00000340
              TIME=TIME + DT                                                     00000350
              ANS=E1P=ANS + 1.                                                   00000360
              DO 20 I=1,NC                                                       00000370
              J=I                                                                00000380
           20 CALL FILT(J)                                                       00000390
      C                                                                          00000400
      C       SENSITIVITIES                                                      00000410
      C                                                                          00000420
              DO 30 I=1,NP                                                       00000430
              J=I                                                                00000440
           30 CALL SENS(JS,J)                                                    00000450
      C                                                                          00000460
      C       LIKELIHOOD ACCUMULATION                                            00000470
              CALL ACUM(JS)                                                      00000480
      C                                                                          00000490
          900 CONTINUE                                                           00000500
      C                                                                          00000510
      C                                                                          00000520
      C       BRANCH TO LOW RATE OPERATIONS ************************************ 00000530
      C                                                                          00000540
              GO TO (910,920,930,940,950), MODE                                  00000550
      C                                                                          00000560
      C       CYCLE 1.  MIN-L CHANNEL SELECTION                                  00000570
      C                                                                          00000580
          910 CONTINUE                                                           00000590
```

```
1642T 01   07-16-79   14.245                                   LABEL PCMLE   PAGE   2

                 CALL CYC1                                              00000600
          C                                                             00000610
                 MODE = 2                                               00000620
                 RETURN                                                 00000630
          C                                                             00000640
          C      CYCLE 2.  SIGNIFICANCE TESTS AND CHANGE LOGIC          00000650
          C                                                             00000660
            920  CONTINUE                                               00000670
                 CALL CYC2                                              00000680
          C                                                             00000690
                 MODE= 3                                                00000700
                 RETURN                                                 00000710
          C                                                             00000720
          C      CHANNEL TRANSFERS                                      00000730
          C                                                             00000740
            930  CONTINUE                                               00000750
                 IF(CHCHAN.EQ.0) GO TO 935                              00000760
                 CALL CYC3                                              00000770
          C                                                             00000780
            935  MODE=4                                                 00000790
                 RETURN                                                 00000800
          C                                                             00000810
          C      CYCLE 4.  PARAMETER INCREMENTS                         00000820
          C                                                             00000830
            940  CONTINUE                                               00000840
                 IF(NOEST) GO TO 945                                    00000850
                 CALL CYC4                                              00000860
          C                                                             00000870
            945  MODE=5                                                 00000880
                 RETURN                                                 00000890
          C                                                             00000900
          C      CYCLE 5.  PARAMETER UPDATES                            00000910
          C                                                             00000920
            950  CONTINUE                                               00000930
                 CALL CYC5                                              00000940
          C                                                             00000950
                 MODE= 1                                                00000960
                 RETURN                                                 00000970
          C                                                             00000980
                 END                                                    00000990
```

HONEYWELL PAGE PRINTING SYSTEM- P1168-02

```
1642T 01   07-16-79   14.245                                        LABEL FILT    PAGE     1

             SUBROUTINE FILT(K)                                                  00001000
             COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS    00001010
             COMMON/MEAS/Y(3),YP(3),XY(2,3),DT,TIME,MODE                         00001020
       C     K=CHANNEL INDEX                                                     00001030
       C     SAVE X                                                              00001040
             X(K,1)=X(K,5)                                                       00001050
             X(K,2)=X(K,6)                                                       00001060
             X(K,3)=X(K,7)                                                       00001070
             X(K,4)=X(K,8)                                                       00001080
       C     RESIDUALS                                                           00001090
             X(K,9)= YP(1) - X(K,1)                                              00001100
             X(K,10)=YP(2) - D(K,14)*X(K,1)-D(K,15)*X(K,2)-D(K,16)*X(K,4)        00001110
       C     UPDATE X                                                            00001120
             X(K,5)=D(K,1)*X(K,1) + D(K,2)*X(K,2) + D(K,3)*X(K,3) +              00001130
            1        D(K,4)*X(K,4) + D(K,11)*YP(3) +                             00001140
            2        F(K,8)*X(K,9) + F(K,10)*X(K,10)                             00001150
             X(K,6)=D(K,5)*X(K,1) + D(K,6)*X(K,2) + D(K,7)*X(K,3) +              00001160
            1        D(K,8)*X(K,4) + D(K,12)*YP(3) +                             00001170
            2        F(K,7)*X(K,9) + F(K,11)*X(K,10)                             00001180
             X(K,7)=D(K,9)*X(K,3) + F(K,8)*X(K,9) + F(K,12)*X(K,10)              00001190
             X(K,8)=D(K,10)*X(K,4) + D(K,13)*YP(3) +                             00001200
            1        F(K,9)*X(K,9) + F(K,13)*X(K,10)                             00001210
       C     (NU)(RI)(NU)                                                        00001220
             TEMP1=X(K,9)*X(K,9)                                                 00001230
             TEMP2=X(K,10)*X(K,10)                                               00001240
             TEMP3=X(K,9)*X(K,10)                                                00001250
             TEMPB=F(K,1)*TEMP1+(F(K,2)+F(K,3))*TEMP3+F(K,4)*TEMP2               00001260
       C     SUM                                                                 00001270
             TJ(K)=E1P*TJ(K) + TEMPB                                             00001280
             RETURN                                                              00001290
             END                                                                 00001300
```

HONEYWELL PAGE PRINTING SYSTEM- P1188-02

```
      SUBROUTINE SENS(K,J)                                                 00001310
      COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS     00001320
      COMMON/MEAS/Y(3),YP(3),XY(2,3),DT,TIME,MODE                          00001330
      COMMON/SENSP/XS(4,10),DS(5,4,16),GE(4),GL(4)                         00001340
     1 ,GSQE(10),GSQL(10),GSQLO(4),GK(5,4,8)                               00001350
C     J=PARAMETER INDEX                                                    00001360
C     K=CHANNEL INDEX                                                      00001370
C     SAVE GRAD X                                                          00001380
      XS(J,1)= XS(J,5)                                                     00001390
      XS(J,2)= XS(J,6)                                                     00001400
      XS(J,3)= XS(J,7)                                                     00001410
      XS(J,4)= XS(J,8)                                                     00001420
C     GRAD NU= -H*(GX)                                                     00001430
      XS(J, 9)= -XS(J,1)                                                   00001440
      XS(J,10)=-D(K,14)*XS(J,1) - D(K,15)*XS(J,2) - D(K,16)*XS(J,4)        00001450
     1      -DS(K,J,14)*X(K,1) - DS(K,J,15)*X(K,2) - DS(K,J,16)*X(K,4)     00001460
C     GRAD X UPDATE                                                        00001470
      XS(J,5)=D(K, 1)*XS(J,1) + D(K, 2)*XS(J,2) + D(K, 3)*XS(J,3) +        00001480
     1       D(K, 4)*XS(J,4) + DS(K,J, 1)*X(K,1) + DS(K,J, 2)*X(K,2) +     00001490
     2       DS(K,J, 3)*X(K,3) + DS(K,J, 4)*X(K,4) + DS(K,J,11)*YP(3) +    00001500
     3      F(K,6)*XS(J,9) + F(K,10)*XS(J,10) +                            00001510
     4      GK(K,J,1)*X(K,9) + GK(K,J,5)*X(K,10)                           00001520
      XS(J,6)=D(K, 5)*XS(J,1) + D(K, 6)*XS(J,2) + D(K, 7)*XS(J,3) +        00001530
     1       D(K, 8)*XS(J,4) + DS(K,J, 5)*X(K,1) + DS(K,J, 6)*X(K,2) +     00001540
     2      DS(K,J,7)*X(K,3) + DS(K,J,8)*X(K,4) + DS(K,J,12)*YP(3) +       00001550
     3      F(K,7)*XS(J,9) + F(K,11)*XS(J,10) +                            00001560
     4         GK(K,J, 2)*X(K,9) + GK(K,J, 6)*X(K,10)                      00001570
      XS(J,7)=D(K,9)*XS(J,3) + DS(K,J,9)*X(K,3)                            00001580
     1     + F(K, 8)*XS(J,9) + F(K,12)*XS(J,10) +                          00001590
     2        GK(K,J, 3)*X(K,9) + GK(K,J, 7)*X(K,10)                       00001600
      XS(J,8)=D(K,10)*XS(J,4) +                                            00001610
     1       F(K,9)*XS(J,9) + F(K,13)*XS(J,10) +                           00001620
     2        GK(K,J, 4)*X(K,9) + GK(K,J, 8)*X(K,10)                       00001630
      RETURN                                                               00001640
      END                                                                  00001650
```

```
      SUBROUTINE ACUM(K)
      COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSO,ANS
      COMMON/SENSP/XS(4,10),DS(5,4,16),GE(4),GL(4)
     1 ,GSQE(10),GSQL(10),GSQLO(4),GK(5,4,8)
C
C     K=CHANNEL INDICATOR
      T11=XS(1,9)*F(K,1) + XS(1,10)*F(K,2)
      T12=XS(1,9)*F(K,3) + XS(1,10)*F(K,4)
      T21=XS(2,9)*F(K,1) + XS(2,10)*F(K,2)
      T22=XS(2,9)*F(K,3) + XS(2,10)*F(K,4)
      T31=XS(3,9)*F(K,1) + XS(3,10)*F(K,2)
      T32=XS(3,9)*F(K,3) + XS(3,10)*F(K,4)
      T41=XS(4,9)*F(K,1) + XS(4,10)*F(K,2)
      T42=XS(4,9)*F(K,3) + XS(4,10)*F(K,4)
C
C     S=(GRAD NU)(RI)(NU)
C
      S1=T11*X(K,9) + T12*X(K,10)
      S2=T21*X(K,9) + T22*X(K,10)
      S3=T31*X(K,9) + T32*X(K,10)
      S4=T41*X(K,9) + T42*X(K,10)
      GE(1)=E2P*(GE(1) - S1) + S1
      GE(2)=E2P*(GE(2) - S2) + S2
      GE(3)=E2P*(GE(3) - S3) + S3
      GE(4)=E2P*(GE(4) - S4) + S4
C
      GL(1)= E1P*GL(1) + GE(1)
      GL(2)= E1P*GL(2) + GE(2)
      GL(3)= E1P*GL(3) + GE(3)
      GL(4)= E1P*GL(4) + GE(4)
C
C     S = (GRAD NU)(RI)(GRAD NU)
C
      S11=T11*XS(1,9) + T12*XS(1,10)
      S12=T11*XS(2,9) + T12*XS(2,10)
      S13=T11*XS(3,9) + T12*XS(3,10)
      S14=T11*XS(4,9) + T12*XS(4,10)
      S22=T21*XS(2,9) + T22*XS(2,10)
      S23=T21*XS(3,9) + T22*XS(3,10)
      S24=T21*XS(4,9) + T22*XS(4,10)
      S33=T31*XS(3,9) + T32*XS(3,10)
      S34=T31*XS(4,9) + T32*XS(4,10)
      S44=T41*XS(4,9) + T42*XS(4,10)
C
      GSQE(1)=E2P*(GSQE(1) - S11) + S11
      GSQE(2)=E2P*(GSQE(2) - S12) + S12
      GSQE(3)=E2P*(GSQE(3) - S13) + S13
      GSQE(4)=E2P*(GSQE(4) - S14) + S14
      GSQE(5)=E2P*(GSQE(5) - S22) + S22
      GSQE(6)=E2P*(GSQE(6) - S23) + S23
      GSQE(7)=E2P*(GSQE(7) - S24) + S24
      GSQE(8)=E2P*(GSQE(8) - S33) + S33
      GSQE(9)=E2P*(GSQE(9) - S34) + S34
```

```
          GSQE(10)=E2P*(GSQE(10) - S44) + S44                           00002190
    C                                                                   00002200
          GSQL(1) = E1P*GSQL(1) + GSQE(1)                               00002210
          GSQL(2) = E1P*GSQL(2) + GSQE(2)                               00002220
          GSQL(3) = E1P*GSQL(3) + GSQE(3)                               00002230
          GSQL(4) = E1P*GSQL(4) + GSQE(4)                               00002240
          GSQL(5) = E1P*GSQL(5) + GSQE(5)                               00002250
          GSQL(6) = E1P*GSQL(6) + GSQE(6)                               00002260
          GSQL(7) = E1P*GSQL(7) + GSQE(7)                               00002270
          GSQL(8) = E1P*GSQL(8) + GSQE(8)                               00002280
          GSQL(9) = E1P*GSQL(9) + GSQE(9)                               00002290
          GSQL(10)= E1P*GSQL(10)+ GSQE(10)                              00002300
          RETURN                                                        00002310
          END                                                           00002320
```

1642T 01   07-16-79   14.247   LABEL FH   PAGE   1

```
      SUBROUTINE FH(I)
C
      COMMON/FHDAT/D,W,C1,C2
      COMMON/MEAS/Y(3),YP(3),XY(2,3),DT,TIME,MODE
C     I=INDEX FOR MEAS
C     FILTER
C     Y/U= (S*S)/(S*S + 2*D*W*S + W*W)
C
      YP(I)=Y(I) - XY(2,I)*C2 - XY(1,I)*C1
      XY(1,I)=XY(1,I) + DT*XY(2,I)
      XY(2,I)=XY(2,I) + DT*YP(I)
      RETURN
      END
```

HONEYWELL PAGE PRINTING SYSTEM- P1189-03

1642T 01 07-18-79 14.247   LABEL CYC1   PAGE   1

```
      SUBROUTINE CYC1                                                   00002460
      COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS  00002470
      COMMON/PARM/ZP(5,4),DZP(4),ZP1,ZP2,ZP3,ZP4,Z1MIN                  00002480
     1 ,NP,NC,JSTEMP,JS,CHCHAN                                          00002490
      TLMAX = -1.E10                                                    00002500
      TLMIN = 1.E10                                                     00002510
      TJMIN=1.E10                                                       00002520
      DO 911 I=1,NC                                                     00002530
      S1=TJ(I) + SIGSQ*ANS*F(I,5)                                       00002540
      TL(I)= S1                                                         00002550
      IF(S1.GT.TLMAX) TLMAX=S1                                          00002560
      IF(S1.LT.TLMIN) TLMIN=S1                                          00002570
      IF(TJ(I).GT.TJMIN) GO TO 911                                      00002580
      TJMIN = TJ(I)                                                     00002590
      JSTEMP=I                                                          00002600
  911 CONTINUE                                                          00002610
      RETURN                                                            00002620
      END                                                               00002630
```

```
1842T 01   07-18-79    14.247                                                      LABEL  CYC2     PAGE      1

              SUBROUTINE CYC2                                                      00002640
      C                                                                            00002650
              COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS     00002660
              COMMON/SENSP/XS(4,10),DS(5,4,16),GE(4),GL(4)                         00002670
             1    ,GSQE(10),GSQL(10),GSQLO(4),GK(5,4,8)                            00002680
              COMMON/VARDAT/UX(50),LX(50)                                          00002690
              COMMON/TSW/RTJS,THRTJC,RTJC,THRTJZ,RTJZ                              00002700
              COMMON/PARM/ZP(5,4),DZP(4),ZP1,ZP2,ZP3,ZP4,ZIMIN                     00002710
             1    ,NP,NC,JSTEMP,JS,CHCHAN                                          00002720
              LOGICAL LX                                                           00002730
              INTEGER CHCHAN                                                       00002740
      C                                                                            00002750
      C                                                                            00002760
              CHCHAN=0                                                             00002770
              TJE=ZP(JS,1)**2*GSQL(1)                                              00002780
      C                                                                            00002790
              IF(TJE.GT.RTJS*TJ(JS)) GO TO 925                                     00002800
              SIGSQ= TJ(JS)/(ANS + ANS)                                            00002810
              IF(SIGSQ.GT.2.) SIGSQ=2.                                             00002820
              IF(SIGSQ.LT.1.0E-04) SIGSQ=1.0E-04                                   00002830
      C                                                                            00002840
          925 IF(LX(1)) GO TO 926                                                  00002850
              IF(TL(JS) - TL(JSTEMP).GT.(THRTJC+RTJC*TJE)) CHCHAN=1                00002860
      C                                                                            00002870
          926 CONTINUE                                                             00002880
              DO 927 I=1,NC                                                        00002890
              IF(TL(I) - TL(JS).LT.(THRTJZ + RTJZ*TJE)) ZIMIN=ZP(I,1)              00002900
          927 CONTINUE                                                             00002910
              RETURN                                                               00002920
              END                                                                  00002930
```

```
      SUBROUTINE CYC3
      COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS
      COMMON/SENSP/XS(4,10),DS(5,4,16),GE(4),GL(4)
     1 ,GSQE(10),GSQL(10),GSQLO(4),GK(5,4,8)
      COMMON/PARM/ZP(5,4),DZP(4),ZP1,ZP2,ZP3,ZP4,ZIMIN
     1 ,NP,NC,JSTEMP,JS,CHCHAN
      COMMON/MEAS/Y(3),YP(3),XY(2,3),DT,TIME,MODE
      INTEGER CHCHAN
C
      DO 931 I=1,8
      DO 931 J=1,NP
  931 XS(J,I)=0.
      IF(TIME.GT.5.0) GO TO 9301
      DO 9303 I=1,10
      GSQL(I)=0.
 9303 GSQE(I)=0.
      DO 9302 I=1,4
      DZP(I)=0.
      GL(I)=0.
 9302 GE(I)=0.
      GO TO 934
 9301 CONTINUE
C
C     CHANNEL TRANSFER SEQUENCE (NR)
C     SCALE GRAD SQ L
      S1=1. + ZP(JS,1)/ZP(JSTEMP,1)
      GSQL(1)= GSQL(1)*S1*S1
      GSQL(2)= GSQL(2)*S1
      GSQL(3)= GSQL(3)*S1
      GSQL(4)= GSQL(4)*S1
C
      S1=GSQL(1) + GSQLO(1)
      S2=GSQL(5) + GSQLO(2)
      S3=GSQL(8) + GSQLO(3)
      S4=GSQL(10)+ GSQLO(4)
C     NEW GRAD L
      DZP(1)=-ZP1 + ZP(JSTEMP,1)
      DZP(2)=-ZP2 + ZP(JSTEMP,2)
      DZP(3)=-ZP3 + ZP(JSTEMP,3)
      DZP(4)=-ZP4 + ZP(JSTEMP,4)
      GL(1)=S1      *DZP(1)+GSQL(2)*DZP(2)+GSQL(3)*DZP(3)+GSQL(4)*DZP(4)
      GL(2)=GSQL(2)*DZP(1)+S2      *DZP(2)+GSQL(6)*DZP(3)+GSQL(7)*DZP(4)
      GL(3)=GSQL(3)*DZP(1)+GSQL(6)*DZP(2)+S3      *DZP(3)+GSQL(9)*DZP(4)
      GL(4)=GSQL(4)*DZP(1)+GSQL(7)*DZP(2)+GSQL(9)*DZP(3)+S4      *DZP(4)
  934 JS=JSTEMP
C
      RETURN
      END
```

```
1642T 01  07-16-79   14.246                                           LABEL CYC4    PAGE     1

C                                                                         00003420
      SUBROUTINE CYC4                                                     00003430
      COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS    00003440
      COMMON/SENSP/XS(4,10),DS(5,4,16),QE(4),QL(4)                        00003450
     1  ,GSQE(10),GSQL(10),GSQLO(4),GK(5,4,8)                             00003460
      COMMON/PARM/ZP(5,4),DZP(4),ZP1,ZP2,ZP3,ZP4,ZIMIN                    00003470
     1  ,NP,NC,JSTEMP,JS,CHCHAN                                           00003480
      EQUIVALENCE(R2,GSQL(2)),(R3,GSQL(3)),(R4,GSQL(4))                   00003490
      EQUIVALENCE(R6,GSQL(6)),(R7,GSQL(7)),(R9,GSQL(9))                   00003500
C     NR INCREMENTS                                                       00003510
      R1=GSQL(1) + GSQLO(1)                                               00003520
      R5=GSQL(5) + GSQLO(2)                                               00003530
      R8=GSQL(8) + GSQLO(3)                                               00003540
      R10=GSQL(10)+GSQLO(4)                                               00003550
C     COFACTORS FOR EXPLICIT INVERSE                                      00003560
      R81099=R8*R10-R9*R9                                                 00003570
      R61097=R6*R10-R9*R7                                                 00003580
      R6978= R6*R9-R7*R8                                                  00003590
      R3764=R3*R7-R6*R4                                                   00003600
      R2754=R2*R7-R5*R4                                                   00003610
      R2653=R2*R6-R5*R3                                                   00003620
      R21047=R2*R10 -R4*R7                                                00003630
      R2937=R2*R9 -R3*R7                                                  00003640
      R2836=R2*R8-R3*R6                                                   00003650
C     INVERSE                                                             00003660
      C1= R5*R81099 - R6*R61097 + R7*R6978                                00003670
      C2=-(R2*R81099 - R3*R61097 + R4*R6978)                              00003680
      C3=R7*R3764 - R9*R2754 + R10*R2653                                  00003690
      C4=-(R6*R3764 - R8*R2754 + R9*R2653)                                00003700
      C5=R1*R81099 - R3*(R3*R10 - R4*R9) + R4*(R3*R9 - R4*R8)             00003710
      C6=-(R1*R61097 - R3*R21047 + R4*(R2*R9 - R4*R6))                    00003720
      C7=R1*R6978 - R3*R2937 + R4*R2836                                   00003730
      C8=R1*(R5*R10 - R7*R7) -R2*R21047 + R4*R2754                        00003740
      C9=-(R1*(R5*R9 - R6*R7) - R2*R2937 + R4*R2653)                      00003750
      C10=R1*(R5*R8 - R6*R6) - R2*R2836 + R3*R2653                        00003760
      DET= R1*C1 + R2*C2 + R3*C3 + R4*C4                                  00003770
      IF(DET.EQ.0.0) STOP 21                                              00003780
C     INCREMENTS                                                          00003790
      DZP(1)= -(C1*GL(1)+C2*GL(2)+C3*GL(3)+C4*GL(4))/DET                  00003800
      DZP(2)= -(C2*GL(1)+C5*GL(2)+C6*GL(3)+C7*GL(4))/DET                  00003810
      DZP(3)= -(C3*GL(1)+C6*GL(2)+C8*GL(3)+C9*GL(4))/DET                  00003820
      DZP(4)= -(C4*GL(1)+C7*GL(2)+C9*GL(3)+C10*GL(4))/DET                 00003830
      RETURN                                                              00003840
      END                                                                 00003850
```

```
      SUBROUTINE CYC5                                                   00003860
C                                                                       00003870
      COMMON/PARM/ZP(5,4),DZP(4),ZP1,ZP2,ZP3,ZP4,Z1MIN                  00003880
     1  ,NP,NC,JSTEMP,JS,CHCHAN                                         00003890
      COMMON/PLIM/ZP1MAX,ZP2MAX,ZP3MAX,ZP4MAX,ZP1MIN,ZP2MIN,ZP3MIN,     00003900
     1  ZP4MIN                                                          00003910
C     UPDATE ZP                                                         00003920
      ZP1= ZP(JS,1) + DZP(1)                                            00003930
      ZP2= ZP(JS,2) + DZP(2)                                            00003940
      ZP3= ZP(JS,3) + DZP(3)                                            00003950
      ZP4= ZP(JS,4) + DZP(4)                                            00003960
      IF(ZP1.GT.ZP1MAX) ZP1=ZP1MAX                                      00003970
      IF(ZP2.GT.ZP2MAX) ZP2=ZP2MAX                                      00003980
      IF(ZP3.GT.ZP3MAX) ZP3=ZP3MAX                                      00003990
      IF(ZP4.GT.ZP4MAX) ZP4=ZP4MAX                                      00004000
      IF(ZP1.LT.ZP1MIN) ZP1=ZP1MIN                                      00004010
      IF(ZP2.LT.ZP2MIN) ZP2=ZP2MIN                                      00004020
      IF(ZP3.LT.ZP3MIN) ZP3=ZP3MIN                                      00004030
      IF(ZP4.LT.ZP4MIN) ZP4=ZP4MIN                                      00004040
C     OTHER VARIABLES                                                   00004050
      RETURN                                                            00004060
      END                                                               00004070
                                                                        00004080
```

HONEYWELL PAGE PRINTING SYSTEM- P1155-02

APPENDIX B

TI990 ARITHMETIC MODES

The fundamental limitation on the execution speed of the PCMLE algorithm is the time required to perform a product and accumulation (PA). The FORTRAN compiler for the TI990 allows for different arithmetic modes, offering tradeoffs with respect to speed, precision, and simplicity of use (see Table B-1). To establish a baseline, the algorithm was executed in its original form, which was heavily dependent on floating-point arithmetic. The unacceptable total execution time of 420 ms/cycle (two parameters, five channels) reflects the floating-point PA time of 1400 ms. The remaining three modes were then considered.

Fixed-integer arithmetic appeared the most promising, combining the speed of the integer mode with the automatic scale factor adjustment. Unfortunately, fixed integer was not available in an extended (double-precision) form, thus limiting products to one word (16 bits) and variables to eight bits (since greater values would induce multiplication overflow). Eight bits (two to three decimal digits) does not provide the required accuracy, so this alternative was eliminated.

Next extended-integer operation was examined. The algorithm was recoded by replacing many of the floating-point PAs with extended-integer (32 bits) operations, explicitly scaled with multiplication or division by an appropriate power of two. This change was expected to provide a significant decrease in execution time, but yielded a surprisingly high 300 ms/cycle. After-the-fact analysis revealed a PA time (including scaling) of 1240 ms. This can be explained by the fact that extended-integer operations are performed via a call to a complex subroutine rather than being compiled in-line. With the addition of the explicit scaling operation, the time is essentially doubled, yielding the aforementioned value for the extended-integer PA.

TABLE B-1. COMPARISON OF ALTERNATE METHODS OF PRODUCT/ACCUMULATION (TI990)

Type Characteristics	Speed (ms)	Precision	Simplicity of Use
Real	1400	8-bit exponent 24-bit mantissa	Automatic scaling
Fixed-integer	60	16 bits; only 8 can be used (to avoid overflow)	Automatic scaling performed in-line
Extended-integer	1240	32 bits; only 16 can be used (to avoid overflow)	Explicit scaling performed in-line
Integer	180	16 bits; 16 can be used with appropriate assembly language modification	Explicit scaling performed via subroutine call

The final recourse was to explore the possibilities of integer (16-bit) arithmetic. In the TI990, a multiplication of two 16-bit integers yields a two-word (32-bit) unsigned result. The FORTRAN compiler uses the lower-order word of the product which can, in most applications (sufficiently small operands), be considered as a signed one-word result. The PCMLE requirements are slightly different, however. It is necessary to multiply two large signed integers and return a scaled (shifted) version of the higher-order word (which contains the most significant bits of the product). This need can most easily be visualized by considering the operands to be fractions. The result would then be a two-word fraction, of which only the higher-order one would be significant. In order to access this higher-order word, it is necessary either to:

1. Modify the compiler output
2. Write FORTRAN-callable assembly language subroutines

In addition to being impractical, the first option would require too much memory. The second option, because it introduces the added time of a general subroutine interface, also seems undesirable. However, upon careful consideration of the specific requirements with respect to number of arguments and addressing mode, the overhead can be drastically reduced. The second recoding replaced all the multiplications in subroutines FILT, SENS, and ACUM with a call to one of four functions (see Table B-2). Function S performs the basic multiplication and scaling operations and its speed determines the PA time of 180 ms. This significant reduction in PA time accounts for a measured average cycle time of 77 ms (two parameter, five channels) with a possible reduction of 7.5 ms per channel eliminated.

TABLE B-2. FUNDAMENTAL OPERATIONS DEFINITIONS

Calling Sequence	Function
S(I*, J*, Q*)	Multiplies two signed integers and returns the higher-order word shifted by Q bits (Q > 0 left, Q < 0 right)
MS(I, J, Q*)	Multiplies two signed integers and returns their two-word product shifted by Q bits
SD(X, Q*)	Shifts a signed extended-integer (two words) by Q bits
MD(I, X)	Multiplies an unsigned integer by a signed extended integer and returns the two highest-order words

* The subprogram assumes indirect addressing on these arguments.

APPENDIX C

TIPCMLE-FORTRAN LISTING

```
      SUBROUTINE PCMLE
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
C
C     HIGH PASS INPUTS
C
      CALL FH(1)
      CALL FH(2)
      CALL FH(3)
C
      IF(.NOT.MLE) RETURN
C
      TIME = TIME + DT
      ANS = MD(E1P,ANS) + ANSI
C
C     STATE UPDATES
C
      DO 10 I=1,NC
   10 CALL FILT(I)
C
C     SENSITIVITY UPDATES
C
      DO 20 I=1,NP
   20 CALL SENS(JS,I)
C
C     LIKELIHOOD ACCUMULATION
C
      CALL ACUM(JS)
C
C     BRANCH TO LOW RATE OPERATIONS *********************************
C
      GO TO (30,40,50,60,70), MODE
C
C     CYCLE 1.  MIN-L CHANNEL SELECTION
C
   30 CALL CYC1
C
      MODE = 2
      RETURN
C
C     CYCLE 2.  SIGNIFICANCE TESTS AND CHANGE LOGIC
C
```

```
   40 CALL CYC2
C
      MODE = 3
      RETURN
C
C     CYCLE 3.  CHANNEL TRANSFERS
C
   50 IF(CHC.AND.CHCHAN) CALL CYC3
C
      MODE = 4
      RETURN
C
C     CYCLE 4.  PARAMETER INCREMENTS
C
   60 IF(EST) CALL CYC4
C
      MODE = 5
      RETURN
C
C     CYCLE 5.  PARAMETER UPDATES
C
   70 CALL CYC5
C
      MODE = 1
      RETURN
C
      END
```

```
COMMON BLOCK/LOGL   / ALLOCATION  0008 BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0000 CHC     LOGICAL         2 SCALAR    0002 CHCHAN LOGICAL          2 SCALAR
0004 EST     LOGICAL         2 SCALAR    0006 MLE    LOGICAL          2 SCALAR

COMMON BLOCK/INT2   / ALLOCATION  046A BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0000 C1HP    INTEGER*2       2 SCALAR    0002 C2HP    INTEGER*2       2 SCALAR
0004 D       INTEGER*2     160 ARRAY     00A4 DS      INTEGER*2     320 ARRAY
01E4 E1P     INTEGER*2       2 SCALAR    01E6 E2P     INTEGER*2       2 SCALAR
01E8 F       INTEGER*2     130 ARRAY     026A GK      INTEGER*2     160 ARRAY
030A JS      INTEGER*2       2 SCALAR    030C JSTEMP INTEGER*2        2 SCALAR
030E MODE    INTEGER*2       2 SCALAR    0310 NC      INTEGER*2       2 SCALAR
0312 NP      INTEGER*2       2 SCALAR    0314 Q       INTEGER*2     176 ARRAY
03C4 SGANS   INTEGER*2       2 SCALAR    03C6 X       INTEGER*2     100 ARRAY
042A XS      INTEGER*2      40 ARRAY     0452 XY      INTEGER*2      12 ARRAY
045E Y       INTEGER*2       6 ARRAY     0464 YP      INTEGER*2       6 ARRAY

COMMON BLOCK/INT4   / ALLOCATION  0060 BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0000 ANS     INTEGER*4       4 SCALAR    0004 ANSI    INTEGER*4       4 SCALAR
0008 DT      INTEGER*4       4 SCALAR    000C GE      INTEGER*4       8 ARRAY
0014 GL      INTEGER*4       8 ARRAY     001C GSQE    INTEGER*4      12 ARRAY
0028 GSQL    INTEGER*4      12 ARRAY     0034 TJ      INTEGER*4      20 ARRAY
0048 TIME    INTEGER*4       4 SCALAR    004C TL      INTEGER*4      20 ARRAY

COMMON BLOCK/REAL   / ALLOCATION  0144 BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0000 DZP     REAL            8 ARRAY     0008 GSQLO   REAL            8 ARRAY
0010 RTJC    REAL            4 SCALAR    0014 RTJS    REAL            4 SCALAR
0018 RTJZ    REAL            4 SCALAR    001C SZP     REAL          100 ARRAY
0080 SZP2    REAL          100 ARRAY     00E4 THRTJC REAL             4 SCALAR
00E8 THRTJZ REAL             4 SCALAR    00EC ZP      REAL           40 ARRAY
0114 ZPS     REAL           20 ARRAY     0128 ZP1     REAL            4 SCALAR
012C ZP1MAX REAL             4 SCALAR    0130 ZP1MIN REAL             4 SCALAR
0134 ZP2     REAL            4 SCALAR    0138 ZP2MAX REAL             4 SCALAR
013C ZP2MIN REAL             4 SCALAR    0140 Z1MIN   REAL            4 SCALAR


SCALAR ALLOCATION

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0030 I       INTEGER*2       2 SCALAR
```

SUBPROGRAMS CALLED

NAME	TYPE	ARGS	NAME	TYPE	ARGS	NAME	TYPE	ARGS
FH	REAL	1	MD	INTEGER*4	2	FILT	REAL	1
SENS	REAL	2	ACUM	REAL	1	F$RCGO	RUNTIME	
CYC1	REAL	0	CYC2	REAL	0	CYC3	REAL	0
CYC4	REAL	0	CYC5	REAL	0	F$RGMY	RUNTIME	
F$REL	RUNTIME		F$REA	RUNTIME				

STATEMENT LABELS

LOCN	LABEL	USE	LOCN	LABEL	USE	LOCN	LABEL	USE
0062	10	DO END	007C	20	DO END	00AC	30	
00BA	40		00C8	50		00E2	60	
00F6	70		0030	M7		0030	M8	
0030	M9		0062	M10		007C	M11	
00DA	M12		00DA	M13		00DA	M14	
00DA	M15		00DA	M16		00EE	M17	
00EE	M18		00EE	M19				

STATEMENT LOCATIONS

LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN
1	0000	3	0010	4	0010	7	0010	9	0010	12	0010
13	0010	14	0010	18	0010	19	0018	20	0020	22	0028
24	0030	25	0044	29	005C	30	0062	34	0076	35	007C
39	0092	43	009A	47	00AC	49	00B2	50	00B8	54	00BA
56	00C0	57	00C6	61	00C8	63	00DA	64	00E0	68	00E2
70	00EE	71	00F4	75	00F6	77	00FC	78	0102	80	0104

ENTRY=0004
PROGRAM SIZE=010E BYTES
DATA SIZE=0032 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS

```
      SUBROUTINE FILT(K)
C
C     K = CHANNEL INDEX
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
      INTEGER F1,F2,F4
      INTEGER*4 TEMPA,TJK
      DATA I/3/
C
C     SAVE X
C
      X(K,1) = X(K,5)
      X(K,2) = X(K,6)
      X(K,3) = X(K,7)
      X(K,4) = X(K,8)
C
C     RESIDUALS
C
      X(K,9) = YP(1) - X(K,1)
      X(K,10) = YP(2) - S(D(K,14),X(K,1),Q(84))
     1         - S(D(K,15),X(K,2),Q(2)) - S(D(K,16),X(K,4),Q(3))
C
C     UPDATE X
C
      X(K,5) = S(D(K,1),X(K,1),Q(4)) + S(D(K,2),X(K,2),Q(5))
     1         + S(D(K,3),X(K,3),Q(6)) + S(D(K,4),X(K,4),Q(7))
     2         + S(D(K,11),YP(I),Q(8))
     3         + S(F(K,6),X(K,9),Q(9)) + S(F(K,10),X(K,10),Q(10))
      X(K,6) = S(D(K,5),X(K,1),Q(11)) + S(D(K,6),X(K,2),Q(12))
     1         + S(D(K,7),X(K,3),Q(13)) + S(D(K,8),X(K,4),Q(14))
     2         + S(D(K,12),YP(I),Q(15))
     3         + S(F(K,7),X(K,9),Q(16)) + S(F(K,11),X(K,10),Q(17))
      X(K,7) = S(D(K,9),X(K,3),Q(18))
     1         + S(F(K,8),X(K,9),Q(19)) + S(F(K,12),X(K,10),Q(20))
      X(K,8) = S(D(K,10),X(K,4),Q(21)) + S(D(K,13),YP(I),Q(22))
     1         + S(F(K,9),X(K,9),Q(23)) + S(F(K,13),X(K,10),Q(24))
C
C     (NU)(RI)(NU)
C
      ITEMP1 = S(X(K,9),X(K,9),Q(88))
      ITEMP2 = S(X(K,9),X(K,10),Q(88))
      ITEMP4 = S(X(K,10),X(K,10),Q(88))
```

```
      F1 = F(K,1)
      F2 = F(K,2)
      F4 = F(K,4)
      TEMPA = MS(F1,ITEMP1,Q(25)) + MS(F2,ITEMP2,Q(26))
     1        + MS(F4,ITEMP4,Q(27))
C
C     SUM
C
      TJK = TJ(K)
      TJ(K) = MD(E1P,TJK) + SD(TEMPA,Q(28))
C
      RETURN
      END
```

COMMON BLOCK/LOGL / ALLOCATION 0008 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	CHC	LOGICAL	2	SCALAR	0002	CHCHAN	LOGICAL	2	SCALAR
0004	EST	LOGICAL	2	SCALAR	0006	MLE	LOGICAL	2	SCALAR

COMMON BLOCK/INT2 / ALLOCATION 046A BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	C1HP	INTEGER*2	2	SCALAR	0002	C2HP	INTEGER*2	2	SCALAR
0004	D	INTEGER*2	160	ARRAY	00A4	DS	INTEGER*2	320	ARRAY
01E4	E1P	INTEGER*2	2	SCALAR	01E6	E2P	INTEGER*2	2	SCALAR
01E8	F	INTEGER*2	130	ARRAY	026A	GK	INTEGER*2	160	ARRAY
030A	JS	INTEGER*2	2	SCALAR	030C	JSTEMP	INTEGER*2	2	SCALAR
030E	MODE	INTEGER*2	2	SCALAR	0310	NC	INTEGER*2	2	SCALAR
0312	NP	INTEGER*2	2	SCALAR	0314	Q	INTEGER*2	176	ARRAY
03C4	SGANS	INTEGER*2	2	SCALAR	03C6	X	INTEGER*2	100	ARRAY
042A	XS	INTEGER*2	40	ARRAY	0452	XY	INTEGER*2	12	ARRAY
045E	Y	INTEGER*2	6	ARRAY	0464	YP	INTEGER*2	6	ARRAY

COMMON BLOCK/INT4 / ALLOCATION 0060 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	ANS	INTEGER*4	4	SCALAR	0004	ANSI	INTEGER*4	4	SCALAR
0008	DT	INTEGER*4	4	SCALAR	000C	GE	INTEGER*4	8	ARRAY
0014	GL	INTEGER*4	8	ARRAY	001C	GSQE	INTEGER*4	12	ARRAY
0028	GSQL	INTEGER*4	12	ARRAY	0034	TJ	INTEGER*4	20	ARRAY
0048	TIME	INTEGER*4	4	SCALAR	004C	TL	INTEGER*4	20	ARRAY

COMMON BLOCK/REAL / ALLOCATION 0144 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	DZP	REAL	8	ARRAY	0008	GSQLO	REAL	8	ARRAY
0010	RTJC	REAL	4	SCALAR	0014	RTJS	REAL	4	SCALAR
0018	RTJZ	REAL	4	SCALAR	001C	SZP	REAL	100	ARRAY
0080	SZP2	REAL	100	ARRAY	00E4	THRTJC	REAL	4	SCALAR
00E8	THRTJZ	REAL	4	SCALAR	00EC	ZP	REAL	40	ARRAY
0114	ZPS	REAL	20	ARRAY	0128	ZP1	REAL	4	SCALAR
012C	ZP1MAX	REAL	4	SCALAR	0130	ZP1MIN	REAL	4	SCALAR
0134	ZP2	REAL	4	SCALAR	0138	ZP2MAX	REAL	4	SCALAR
013C	ZP2MIN	REAL	4	SCALAR	0140	Z1MIN	REAL	4	SCALAR

SCALAR ALLOCATION

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0030	I	INTEGER*2	2	SCALAR	0032	ITEMP1	INTEGER*2	2	SCALAR
0034	ITEMP2	INTEGER*2	2	SCALAR	0036	ITEMP4	INTEGER*2	2	SCALAR

```
0038 F1      INTEGER*2      2 SCALAR   003A F2     INTEGER*2      2 SCALAR
003C F4      INTEGER*2      2 SCALAR   003E TEMPA  INTEGER*4      4 SCALAR
0042 TJK     INTEGER*4      4 SCALAR

DUMMY ARGUMENT ALLOCATION

LOCN NAME    MODE       BYTES TYPE     LOCN NAME   MODE       BYTES TYPE

0046 K       INTEGER*2      2 SCALAR
```

```
SUBPROGRAMS CALLED

NAME     TYPE        ARGS   NAME     TYPE        ARGS   NAME     TYPE        ARGS

S        INTEGER*2 3        MS       INTEGER*4 3        MD       INTEGER*4 2
SD       INTEGER*4 2        F$RGMY   RUNTIME            F$REA    RUNTIME


STATEMENT LOCATIONS

LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN

1    0000    5    0012    6    0012    9    0012    11   0012    14   0012
15   0012    16   0012    17   0012    18   0012    19   0012    23   0012
24   0020    25   002A    26   0034    30   003E    31   004E    36   010C
40   02AE    44   045C    46   0510    51   05FC    52   0634    53   066C
54   06A4    55   06AE    56   06B4    57   06BA    62   0726    63   073A
65   0770    66   0772


ENTRY=0004
PROGRAM SIZE=0772 BYTES
DATA SIZE=00F8 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

```
      SUBROUTINE SENS(K,J)
C
C     J = PARAMETER INDEX
C     K = CHANNEL INDEX
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
      DATA I/3/
C
C     SAVE GRAD X
C
      XS(J,1) = XS(J,5)
      XS(J,2) = XS(J,6)
      XS(J,3) = XS(J,7)
      XS(J,4) = XS(J,8)
C
C     GRAD NU
C
      XS(J,9) = - XS(J,1)
      XS(J,10) = - S(D(K,14),XS(J,1),Q(29)) - S(D(K,15),XS(J,2),Q(30))
     1          - S(D(K,16),XS(J,4),Q(31)) - S(DS(K,J,14),X(K,1),Q(32))
     2          - S(DS(K,J,15),X(K,2),Q(33)) - S(DS(K,J,16),X(K,4),Q(34))
C
C     GRAD X UPDATE
C
      XS(J,5) = S(D(K,1),XS(J,1),Q(35)) + S(D(K,2),XS(J,2),Q(36))
     1         + S(D(K,3),XS(J,3),Q(37)) + S(D(K,4),XS(J,4),Q(38))
     2         + S(DS(K,J,1),X(K,1),Q(39)) + S(DS(K,J,2),X(K,2),Q(40))
     3         + S(DS(K,J,3),X(K,3),Q(41)) + S(DS(K,J,4),X(K,4),Q(42))
     4         + S(DS(K,J,11),YP(I),Q(43))
     5         + S(F(K,6),XS(J,9),Q(44)) + S(F(K,10),XS(J,10),Q(45))
     6         + S(GK(K,J,1),X(K,9),Q(46)) + S(GK(K,J,5),X(K,10),Q(47))
      XS(J,6) = S(D(K,5),XS(J,1),Q(48)) + S(D(K,6),XS(J,2),Q(49))
     1         + S(D(K,7),XS(J,3),Q(50)) + S(D(K,8),XS(J,4),Q(51))
     2         + S(DS(K,J,5),X(K,1),Q(52)) + S(DS(K,J,6),X(K,2),Q(53))
     3         + S(DS(K,J,7),X(K,3),Q(54)) + S(DS(K,J,8),X(K,4),Q(55))
     4         + S(DS(K,J,12),YP(I),Q(56))
     5         + S(F(K,7),XS(J,9),Q(57)) + S(F(K,11),XS(J,10),Q(58))
     6         + S(GK(K,J,2),X(K,9),Q(59)) + S(GK(K,J,6),X(K,10),Q(60))
      XS(J,7) = S(D(K,9),XS(J,3),Q(61))
     1         + S(F(K,8),XS(J,9),Q(62)) + S(F(K,12),XS(J,10),Q(63))
     2         + S(GK(K,J,3),X(K,9),Q(64)) + S(GK(K,J,7),X(K,10),Q(65))
```

TI TXDS FORTRAN 936873*B 07/12/79   15:22:26   OPTIONS: S   PAGE   2

```
       XS(J,8) = S(D(K,10),XS(J,4),Q(66))
     1         + S(F(K,9),XS(J,9),Q(67)) + S(F(K,13),XS(J,10),Q(68))
     2         + S(GK(K,J,4),X(K,9),Q(69)) + S(GK(K,J,8),X(K,10),Q(70))
C
        RETURN
        END
```

COMMON BLOCK/LOGL / ALLOCATION 0008 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	CHC	LOGICAL	2	SCALAR	0002	CHCHAN	LOGICAL	2	SCALAR
0004	EST	LOGICAL	2	SCALAR	0006	MLE	LOGICAL	2	SCALAR

COMMON BLOCK/INT2 / ALLOCATION 046A BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	C1HP	INTEGER*2	2	SCALAR	0002	C2HP	INTEGER*2	2	SCALAR
0004	D	INTEGER*2	160	ARRAY	00A4	DS	INTEGER*2	320	ARRAY
01E4	E1P	INTEGER*2	2	SCALAR	01E6	E2P	INTEGER*2	2	SCALAR
01E8	F	INTEGER*2	130	ARRAY	026A	GK	INTEGER*2	160	ARRAY
030A	JS	INTEGER*2	2	SCALAR	030C	JSTEMP	INTEGER*2	2	SCALAR
030E	MODE	INTEGER*2	2	SCALAR	0310	NC	INTEGER*2	2	SCALAR
0312	NP	INTEGER*2	2	SCALAR	0314	Q	INTEGER*2	176	ARRAY
03C4	SGANS	INTEGER*2	2	SCALAR	03C6	X	INTEGER*2	100	ARRAY
042A	XS	INTEGER*2	40	ARRAY	0452	XY	INTEGER*2	12	ARRAY
045E	Y	INTEGER*2	6	ARRAY	0464	YP	INTEGER*2	6	ARRAY

COMMON BLOCK/INT4 / ALLOCATION 0060 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	ANS	INTEGER*4	4	SCALAR	0004	ANSI	INTEGER*4	4	SCALAR
0008	DT	INTEGER*4	4	SCALAR	000C	GE	INTEGER*4	8	ARRAY
0014	GL	INTEGER*4	8	ARRAY	001C	GSQE	INTEGER*4	12	ARRAY
0028	GSQL	INTEGER*4	12	ARRAY	0034	TJ	INTEGER*4	20	ARRAY
0048	TIME	INTEGER*4	4	SCALAR	004C	TL	INTEGER*4	20	ARRAY

COMMON BLOCK/REAL / ALLOCATION 0144 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	DZP	REAL	8	ARRAY	0008	GSQLO	REAL	8	ARRAY
0010	RTJC	REAL	4	SCALAR	0014	RTJS	REAL	4	SCALAR
0018	RTJZ	REAL	4	SCALAR	001C	SZP	REAL	100	ARRAY
0080	SZP2	REAL	100	ARRAY	00E4	THRTJC	REAL	4	SCALAR
00E8	THRTJZ	REAL	4	SCALAR	00EC	ZP	REAL	40	ARRAY
0114	ZPS	REAL	20	ARRAY	0128	ZP1	REAL	4	SCALAR
012C	ZP1MAX	REAL	4	SCALAR	0130	ZP1MIN	REAL	4	SCALAR
0134	ZP2	REAL	4	SCALAR	0138	ZP2MAX	REAL	4	SCALAR
013C	ZP2MIN	REAL	4	SCALAR	0140	Z1MIN	REAL	4	SCALAR

SCALAR ALLOCATION

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0030	I	INTEGER*2	2	SCALAR					

```
DUMMY ARGUMENT ALLOCATION

LOCN NAME    MODE        BYTES TYPE     LOCN NAME    MODE        BYTES TYPE

0032 K       INTEGER*2       2 SCALAR   0034 J       INTEGER*2       2 SCALAR
```

SUBPROGRAMS CALLED

NAME	TYPE	ARGS	NAME	TYPE	ARGS	NAME	TYPE	ARGS
S	INTEGER*2	3	F$RGMY	RUNTIME				

STATEMENT LOCATIONS

LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN
1	0000	6	0012	7	0012	10	0012	12	0012	15	0012
16	0012	17	0012	18	0012	22	0012	23	0020	24	002A
25	0034	29	003E	30	0054	36	01D6	43	0568	50	08A2
53	09D6	57	0B0A	58	0B0C						

ENTRY=0004
PROGRAM SIZE=0B0C BYTES
DATA SIZE=0102 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS

```
TI TXDS FORTRAN 936873*B 07/12/79   15:54:04    OPTIONS: S            PAGE   1

      SUBROUTINE ACUM(K)
C
C     K = CHANNEL INDEX
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE1,GE2,GL1,GL2,GSQE1,GSQE2,GSQE3,
     1             GSQL1,GSQL2,GSQL3,TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE1,GE2,GL1,GL2,GSQE1,GSQE2,GSQE3,
     1          GSQL1,GSQL2,GSQL3,MD,MS,SD,TJ,TIME,TL
      INTEGER T11,T12,T21,T22,X9,X10
      INTEGER*4 S1,S2,S11,S12,S22
      EQUIVALENCE (NU11,XS(1,9)),(NU12,XS(1,10)),
     1            (NU21,XS(2,9)),(NU22,XS(2,10))
C
      T11 = S(XS(1,9),F(K,1),Q(87)) + S(XS(1,10),F(K,2),Q(71))
      T12 = S(XS(1,9),F(K,3),Q(72)) + S(XS(1,10),F(K,4),Q(87))
      T21 = S(XS(2,9),F(K,1),Q(87)) + S(XS(2,10),F(K,2),Q(71))
      T22 = S(XS(2,9),F(K,3),Q(72)) + S(XS(2,10),F(K,4),Q(87))
C
C     S = (GRAD NU)(RI)(NU)
C
      X9 = X(K,9)
      X10 = X(K,10)
      S1 = MS(T11,X9,Q(73)) + MS(T12,X10,Q(74))
      S2 = MS(T21,X9,Q(73)) + MS(T22,X10,Q(74))
C
      GE1 = MD(E2P,GE1-S1) + S1
      GE2 = MD(E2P,GE2-S2) + S2
C
      GL1 = MD(E1P,GL1) + SD(GE1,Q(75))
      GL2 = MD(E1P,GL2) + SD(GE2,Q(75))
C
C     S = (GRAD NU)(RI)(GRAD NU)
C
      S11 = MS(T11,NU11,Q(76)) + MS(T12,NU12,Q(77))
      S12 = MS(T11,NU21,Q(76)) + MS(T12,NU22,Q(77))
      S22 = MS(T21,NU21,Q(76)) + MS(T22,NU22,Q(77))
C
      GSQE1 = MD(E2P,GSQE1-S11) + S11
      GSQE2 = MD(E2P,GSQE2-S12) + S12
      GSQE3 = MD(E2P,GSQE3-S22) + S22
C
      GSQL1 = MD(E1P,GSQL1) + SD(GSQE1,Q(78))
      GSQL2 = MD(E1P,GSQL2) + SD(GSQE2,Q(78))
      GSQL3 = MD(E1P,GSQL3) + SD(GSQE3,Q(78))
```

```
TI TXDS FORTRAN 936873*B 07/12/79   15:54:04    OPTIONS: S           PAGE  2

      C
              RETURN
              END
```

```
COMMON BLOCK/LOGL   / ALLOCATION  0008 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE       BYTES TYPE

0000 CHC    LOGICAL         2 SCALAR    0002 CHCHAN LOGICAL        2 SCALAR
0004 EST    LOGICAL         2 SCALAR    0006 MLE    LOGICAL        2 SCALAR

COMMON BLOCK/INT2   / ALLOCATION  046A BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE       BYTES TYPE

0000 C1HP   INTEGER*2       2 SCALAR    0002 C2HP   INTEGER*2      2 SCALAR
0004 D      INTEGER*2     160 ARRAY     00A4 DS     INTEGER*2    320 ARRAY
01E4 E1P    INTEGER*2       2 SCALAR    01E6 E2P    INTEGER*2      2 SCALAR
01E8 F      INTEGER*2     130 ARRAY     026A GK     INTEGER*2    160 ARRAY
030A JS     INTEGER*2       2 SCALAR    030C JSTEMP INTEGER*2      2 SCALAR
030E MODE   INTEGER*2       2 SCALAR    0310 NC     INTEGER*2      2 SCALAR
0312 NP     INTEGER*2       2 SCALAR    0314 Q      INTEGER*2    176 ARRAY
03C4 SGANS  INTEGER*2       2 SCALAR    03C6 X      INTEGER*2    100 ARRAY
042A XS     INTEGER*2      40 ARRAY     0452 XY     INTEGER*2     12 ARRAY
045E Y      INTEGER*2       6 ARRAY     0464 YP     INTEGER*2      6 ARRAY
044A NU11   INTEGER*2       2 SCALAR    044E NU12   INTEGER*2      2 SCALAR
044C NU21   INTEGER*2       2 SCALAR    0450 NU22   INTEGER*2      2 SCALAR

COMMON BLOCK/INT4   / ALLOCATION  0060 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE       BYTES TYPE

0000 ANS    INTEGER*4       4 SCALAR    0004 ANSI   INTEGER*4      4 SCALAR
0008 DT     INTEGER*4       4 SCALAR    000C GE1    INTEGER*4      4 SCALAR
0010 GE2    INTEGER*4       4 SCALAR    0014 GL1    INTEGER*4      4 SCALAR
0018 GL2    INTEGER*4       4 SCALAR    001C GSQE1  INTEGER*4      4 SCALAR
0020 GSQE2  INTEGER*4       4 SCALAR    0024 GSQE3  INTEGER*4      4 SCALAR
0028 GSQL1  INTEGER*4       4 SCALAR    002C GSQL2  INTEGER*4      4 SCALAR
0030 GSQL3  INTEGER*4       4 SCALAR    0034 TJ     INTEGER*4     20 ARRAY
0048 TIME   INTEGER*4       4 SCALAR    004C TL     INTEGER*4     20 ARRAY

COMMON BLOCK/REAL   / ALLOCATION  0144 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE       BYTES TYPE

0000 DZP    REAL            8 ARRAY     0008 GSQLO  REAL           8 ARRAY
0010 RTJC   REAL            4 SCALAR    0014 RTJS   REAL           4 SCALAR
0018 RTJZ   REAL            4 SCALAR    001C SZP    REAL         100 ARRAY
0080 SZP2   REAL          100 ARRAY     00E4 THRTJC REAL           4 SCALAR
00E8 THRTJZ REAL            4 SCALAR    00EC ZP     REAL          40 ARRAY
0114 ZPS    REAL           20 ARRAY     0128 ZP1    REAL           4 SCALAR
012C ZP1MAX REAL            4 SCALAR    0130 ZP1MIN REAL           4 SCALAR
0134 ZP2    REAL            4 SCALAR    0138 ZP2MAX REAL           4 SCALAR
013C ZP2MIN REAL            4 SCALAR    0140 Z1MIN  REAL           4 SCALAR
```

SCALAR ALLOCATION

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0030	T11	INTEGER*2	2	SCALAR	0032	T12	INTEGER*2	2	SCALAR
0034	T21	INTEGER*2	2	SCALAR	0036	T22	INTEGER*2	2	SCALAR
0038	X9	INTEGER*2	2	SCALAR	003A	X10	INTEGER*2	2	SCALAR
003C	S1	INTEGER*4	4	SCALAR	0040	S2	INTEGER*4	4	SCALAR
0044	S11	INTEGER*4	4	SCALAR	0048	S12	INTEGER*4	4	SCALAR
004C	S22	INTEGER*4	4	SCALAR					

DUMMY ARGUMENT ALLOCATION

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0050	K	INTEGER*2	2	SCALAR					

```
SUBPROGRAMS CALLED

NAME    TYPE        ARGS   NAME    TYPE        ARGS   NAME    TYPE        ARGS

S       INTEGER*2 3        MS      INTEGER*4 3        MD      INTEGER*4 2
SD      INTEGER*4 2        F$RGMY  RUNTIME            F$REA   RUNTIME
F$REL   RUNTIME            F$RES   RUNTIME

STATEMENT LOCATIONS

LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN

1    0000    5    0012    6    0012    9    0012    11   0012    14   0012
15   0012    16   0012    18   0012    19   0012    20   0012    23   0012
24   0086    25   00F2    26   015E    30   01CA    31   01D4    32   01DA
33   0220    35   0266    36   0292    38   02BE    39   02F4    43   032A
44   0370    45   03B6    47   03FC    48   0428    49   0454    51   0480
52   04B6    53   04EC    55   0522    56   0524

ENTRY=0004
PROGRAM SIZE=0524 BYTES
DATA SIZE=00D2 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

```
      SUBROUTINE FH(I)
C
C     I = MEASUREMENT INDEX
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ CHP(2),D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER CHP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
      INTEGER DTD(2)
      EQUIVALENCE (DT,DTD)
      DATA J/1/,K/2/
C
C     FILTER
C     Y/U= (S*S)/(S*S + 2*D*W*S + W*W)
C
      YP(I) = Y(I) - S(CHP(J),XY(1,I),Q(79)) - S(CHP(K),XY(2,I),Q(80))
      XY(1,I) = XY(1,I) + S(DTD(K),XY(2,I),Q(87))
      XY(2,I) = XY(2,I) + S(DTD(K),YP(I),Q(87))
C
      RETURN
      END
```

```
TI TXDS FORTRAN 936873*B 07/12/79    16:34:14    OPTIONS: S          PAGE   2


COMMON BLOCK/LOGL   / ALLOCATION  0008 BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 CHC     LOGICAL         2 SCALAR    0002 CHCHAN LOGICAL         2 SCALAR
0004 EST     LOGICAL         2 SCALAR    0006 MLE    LOGICAL         2 SCALAR

COMMON BLOCK/INT2   / ALLOCATION  046A BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 CHP     INTEGER*2       4 ARRAY     0004 D      INTEGER*2     160 ARRAY
00A4 DS      INTEGER*2     320 ARRAY     01E4 E1P    INTEGER*2       2 SCALAR
01E6 E2P     INTEGER*2       2 SCALAR    01E8 F      INTEGER*2     130 ARRAY
026A GK      INTEGER*2     160 ARRAY     030A JS     INTEGER*2       2 SCALAR
030C JSTEMP  INTEGER*2       2 SCALAR    030E MODE   INTEGER*2       2 SCALAR
0310 NC      INTEGER*2       2 SCALAR    0312 NP     INTEGER*2       2 SCALAR
0314 Q       INTEGER*2     176 ARRAY     03C4 SGANS  INTEGER*2       2 SCALAR
03C6 X       INTEGER*2     100 ARRAY     042A XS     INTEGER*2      40 ARRAY
0452 XY      INTEGER*2      12 ARRAY     045E Y      INTEGER*2       6 ARRAY
0464 YP      INTEGER*2       6 ARRAY

COMMON BLOCK/INT4   / ALLOCATION  0060 BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 ANS     INTEGER*4       4 SCALAR    0004 ANSI   INTEGER*4       4 SCALAR
0008 DT      INTEGER*4       4 SCALAR    000C GE     INTEGER*4       8 ARRAY
0014 GL      INTEGER*4       8 ARRAY     001C GSQE   INTEGER*4      12 ARRAY
0028 GSQL    INTEGER*4      12 ARRAY     0034 TJ     INTEGER*4      20 ARRAY
0048 TIME    INTEGER*4       4 SCALAR    004C TL     INTEGER*4      20 ARRAY
0008 DTD     INTEGER*2       4 ARRAY

COMMON BLOCK/REAL   / ALLOCATION  0144 BYTES

LOCN NAME    MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 DZP     REAL            8 ARRAY     0008 GSQLO  REAL            8 ARRAY
0010 RTJC    REAL            4 SCALAR    0014 RTJS   REAL            4 SCALAR
0018 RTJZ    REAL            4 SCALAR    001C SZP    REAL          100 ARRAY
0080 SZP2    REAL          100 ARRAY     00E4 THRTJC REAL            4 SCALAR
00E8 THRTJZ  REAL            4 SCALAR    00EC ZP     REAL           40 ARRAY
0114 ZPS     REAL           20 ARRAY     0128 ZP1    REAL            4 SCALAR
012C ZP1MAX  REAL            4 SCALAR    0130 ZP1MIN REAL            4 SCALAR
0134 ZP2     REAL            4 SCALAR    0138 ZP2MAX REAL            4 SCALAR
013C ZP2MIN  REAL            4 SCALAR    0140 Z1MIN  REAL            4 SCALAR
```

```
TI TXDS FORTRAN 936873*B 07/12/79   16:34:14    OPTIONS: S           PAGE   3


SCALAR ALLOCATION

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0030 J       INTEGER*2       2 SCALAR    0032 K       INTEGER*2       2 SCALAR


DUMMY ARGUMENT ALLOCATION

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0034 I       INTEGER*2       2 SCALAR
```

```
SUBPROGRAMS CALLED

NAME      TYPE        ARGS   NAME     TYPE       ARGS   NAME    TYPE      ARGS

S         INTEGER*2 3        F$RGMY   RUNTIME

STATEMENT LOCATIONS

LINE LOCN     LINE LOCN     LINE LOCN     LINE LOCN     LINE LOCN     LINE LOCN

1     0000    5     0012    6     0012    9     0012    11    0012    14    0012
15    0012    16    0012    17    0012    18    0012    19    0012    24    0012
25    0098    26    00D4    28    010C    29    010E

ENTRY=0004
PROGRAM SIZE=010E BYTES
DATA SIZE=0054 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

```
      SUBROUTINE CYC1
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS(1),
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
      INTEGER*4 S1,TJMIN,TLMAX,TLMIN
      DATA J/1/
C
      TJMIN = 2147483647
      TLMAX = -2147483647
      TLMIN = 2147483647
      DO 10 I=1,NC
      S1 = TJ(I) + S(F(I,5),SGANS(J),Q(81))
      TL(I) = S1
      IF(S1.GT.TLMAX) TLMAX = S1
      IF(S1.LT.TLMIN) TLMIN = S1
      IF(TJ(I).GT.TJMIN) GO TO 10
      TJMIN = TJ(I)
      JSTEMP = I
   10 CONTINUE
C
      RETURN
      END
```

COMMON BLOCK/LOGL / ALLOCATION 0008 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	CHC	LOGICAL	2	SCALAR	0002	CHCHAN	LOGICAL	2	SCALAR
0004	EST	LOGICAL	2	SCALAR	0006	MLE	LOGICAL	2	SCALAR

COMMON BLOCK/INT2 / ALLOCATION 046A BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	C1HP	INTEGER*2	2	SCALAR	0002	C2HP	INTEGER*2	2	SCALAR
0004	D	INTEGER*2	160	ARRAY	00A4	DS	INTEGER*2	320	ARRAY
01E4	E1P	INTEGER*2	2	SCALAR	01E6	E2P	INTEGER*2	2	SCALAR
01E8	F	INTEGER*2	130	ARRAY	026A	GK	INTEGER*2	160	ARRAY
030A	JS	INTEGER*2	2	SCALAR	030C	JSTEMP	INTEGER*2	2	SCALAR
030E	MODE	INTEGER*2	2	SCALAR	0310	NC	INTEGER*2	2	SCALAR
0312	NP	INTEGER*2	2	SCALAR	0314	Q	INTEGER*2	176	ARRAY
03C4	SGANS	INTEGER*2	2	ARRAY	03C6	X	INTEGER*2	100	ARRAY
042A	XS	INTEGER*2	40	ARRAY	0452	XY	INTEGER*2	12	ARRAY
045E	Y	INTEGER*2	6	ARRAY	0464	YP	INTEGER*2	6	ARRAY

COMMON BLOCK/INT4 / ALLOCATION 0060 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	ANS	INTEGER*4	4	SCALAR	0004	ANSI	INTEGER*4	4	SCALAR
0008	DT	INTEGER*4	4	SCALAR	000C	GE	INTEGER*4	8	ARRAY
0014	GL	INTEGER*4	8	ARRAY	001C	GSQE	INTEGER*4	12	ARRAY
0028	GSQL	INTEGER*4	12	ARRAY	0034	TJ	INTEGER*4	20	ARRAY
0048	TIME	INTEGER*4	4	SCALAR	004C	TL	INTEGER*4	20	ARRAY

COMMON BLOCK/REAL / ALLOCATION 0144 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	DZP	REAL	8	ARRAY	0008	GSQLO	REAL	8	ARRAY
0010	RTJC	REAL	4	SCALAR	0014	RTJS	REAL	4	SCALAR
0018	RTJZ	REAL	4	SCALAR	001C	SZP	REAL	100	ARRAY
0080	SZP2	REAL	100	ARRAY	00E4	THRTJC	REAL	4	SCALAR
00E8	THRTJZ	REAL	4	SCALAR	00EC	ZP	REAL	40	ARRAY
0114	ZPS	REAL	20	ARRAY	0128	ZP1	REAL	4	SCALAR
012C	ZP1MAX	REAL	4	SCALAR	0130	ZP1MIN	REAL	4	SCALAR
0134	ZP2	REAL	4	SCALAR	0138	ZP2MAX	REAL	4	SCALAR
013C	ZP2MIN	REAL	4	SCALAR	0140	Z1MIN	REAL	4	SCALAR

SCALAR ALLOCATION

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0030	J	INTEGER*2	2	SCALAR	0032	TJMIN	INTEGER*4	4	SCALAR
0036	TLMAX	INTEGER*4	4	SCALAR	003A	TLMIN	INTEGER*4	4	SCALAR

```
003E I        INTEGER*2     2 SCALAR   0040 S1      INTEGER*4     4 SCALAR
```

```
SUBPROGRAMS CALLED

NAME      TYPE        ARGS   NAME      TYPE        ARGS   NAME      TYPE        ARGS

S         INTEGER*2 3        F$RGMY    RUNTIME            F$RITE    RUNTIME
F$REA     RUNTIME            F$RES     RUNTIME            F$RET     RUNTIME
F$REL     RUNTIME

STATEMENT LABELS

LOCN  LABEL USE          LOCN  LABEL USE          LOCN  LABEL USE

0116  10                 00BE  M1                 00DE  M2
003A  M3                 00BE  M4                 00BE  M5
00DE  M6                 00DE  M7                 0116  M8
0116  M9

STATEMENT LOCATIONS

LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN

1    0000    3    0010    4    0010    7    0010    9    0010    12   0010
13   0010    14   0010    15   0010    16   0010    18   0010    19   001C
20   0028    21   0034    22   003A    23   0098    24   00A4    25   00BE
26   00DE    27   00FE    28   0110    29   0116    31   0122    32   0124

ENTRY=0004
PROGRAM SIZE=012E BYTES
DATA SIZE=0052 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

```
      SUBROUTINE CYC2
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
      INTEGER ANSD(2),SGANSM,TJD(2,5)
      INTEGER*4 CSGANS,TEST
      EQUIVALENCE (ANS,ANSD),(TJ,TJD)
C
      TJE = ZPS(JS)*GSQL(1)
      IF(TJE.GT.RTJS*TJ(JS)) GO TO 10
      CSGANS = ISHFT(ANSD(1),Q(82))
      SGANS = ANSD(1)
      IF(TJ(JS).GT.CSGANS) GO TO 10
      SGANSM = ISHFT(ANSD(1),-6)
      SGANS = ISHFT(TJD(2,JS),Q(83))
      IF(SGANS.LT.SGANSM) SGANS = SGANSM
C
   10 CHCHAN = (TL(JS)-TL(JSTEMP)).GT.LFIX(THRTJC+RTJC*TJE)
C
      TEST = TL(JS) + THRTJZ + RTJZ*TJE
      DO 20 I=1,NC
      IF(TL(I).LT.TEST) Z1MIN = ZP(I,1)
   20 CONTINUE
C
      RETURN
      END
```

COMMON BLOCK/LOGL / ALLOCATION 0008 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	CHC	LOGICAL	2	SCALAR	0002	CHCHAN	LOGICAL	2	SCALAR
0004	EST	LOGICAL	2	SCALAR	0006	MLE	LOGICAL	2	SCALAR

COMMON BLOCK/INT2 / ALLOCATION 046A BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	C1HP	INTEGER*2	2	SCALAR	0002	C2HP	INTEGER*2	2	SCALAR
0004	D	INTEGER*2	160	ARRAY	00A4	DS	INTEGER*2	320	ARRAY
01E4	E1P	INTEGER*2	2	SCALAR	01E6	E2P	INTEGER*2	2	SCALAR
01E8	F	INTEGER*2	130	ARRAY	026A	GK	INTEGER*2	160	ARRAY
030A	JS	INTEGER*2	2	SCALAR	030C	JSTEMP	INTEGER*2	2	SCALAR
030E	MODE	INTEGER*2	2	SCALAR	0310	NC	INTEGER*2	2	SCALAR
0312	NP	INTEGER*2	2	SCALAR	0314	Q	INTEGER*2	176	ARRAY
03C4	SGANS	INTEGER*2	2	SCALAR	03C6	X	INTEGER*2	100	ARRAY
042A	XS	INTEGER*2	40	ARRAY	0452	XY	INTEGER*2	12	ARRAY
045E	Y	INTEGER*2	6	ARRAY	0464	YP	INTEGER*2	6	ARRAY

COMMON BLOCK/INT4 / ALLOCATION 0060 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	ANS	INTEGER*4	4	SCALAR	0004	ANSI	INTEGER*4	4	SCALAR
0008	DT	INTEGER*4	4	SCALAR	000C	GE	INTEGER*4	8	ARRAY
0014	GL	INTEGER*4	8	ARRAY	001C	GSQE	INTEGER*4	12	ARRAY
0028	GSQL	INTEGER*4	12	ARRAY	0034	TJ	INTEGER*4	20	ARRAY
0048	TIME	INTEGER*4	4	SCALAR	004C	TL	INTEGER*4	20	ARRAY
0000	ANSD	INTEGER*2	4	ARRAY	0034	TJD	INTEGER*2	20	ARRAY

COMMON BLOCK/REAL / ALLOCATION 0144 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	DZP	REAL	8	ARRAY	0008	GSQLO	REAL	8	ARRAY
0010	RTJC	REAL	4	SCALAR	0014	RTJS	REAL	4	SCALAR
0018	RTJZ	REAL	4	SCALAR	001C	SZP	REAL	100	ARRAY
0080	SZP2	REAL	100	ARRAY	00E4	THRTJC	REAL	4	SCALAR
00E8	THRTJZ	REAL	4	SCALAR	00EC	ZP	REAL	40	ARRAY
0114	ZPS	REAL	20	ARRAY	0128	ZP1	REAL	4	SCALAR
012C	ZP1MAX	REAL	4	SCALAR	0130	ZP1MIN	REAL	4	SCALAR
0134	ZP2	REAL	4	SCALAR	0138	ZP2MAX	REAL	4	SCALAR
013C	ZP2MIN	REAL	4	SCALAR	0140	Z1MIN	REAL	4	SCALAR

```
TI TXDS FORTRAN 936873*B 07/12/79   16:45:15    OPTIONS: S           PAGE   3


SCALAR ALLOCATION

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0030 TJE    REAL            4 SCALAR    0034 CSGANS INTEGER*4       4 SCALAR
0038 SGANSM INTEGER*2       2 SCALAR    003A TEST   INTEGER*4       4 SCALAR
003E I      INTEGER*2       2 SCALAR
```

```
SUBPROGRAMS CALLED

NAME      TYPE        ARGS   NAME      TYPE         ARGS   NAME      TYPE        ARGS

ISHFT     INTEGER*2 2        LFIX   .  INTEGER*4 1         F$RREL    RUNTIME
F$RGMY    RUNTIME            F$REL     RUNTIME             F$RITP    RUNTIME
F$RISH    RUNTIME            F$RITE    RUNTIME             F$RES     RUNTIME
F$RET     RUNTIME

STATEMENT LABELS

LOCN  LABEL  USE          LOCN  LABEL  USE          LOCN  LABEL  USE

00D2  10                  01A2  20     DO END       00D2  M2
00D2  M3                  00D2  M4                  00D2  M5
00D2  M6                  00D2  M7                  00D2  M8
01A2  M9                  0128  M10                 0128  M11
012A  M12                 012A  M13                 016E  M14
01A2  M15                 01A2  M16

STATEMENT LOCATIONS

LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN

1    0000    3    0010    4    0010    7    0010    9    0010    12   0010
13   0010    14   0010    15   0010    16   0010    17   0010    19   0010
20   002A    21   004E    22   0074    23   007A    24   0094    25   009E
26   00C4    28   00D2    30   012E    31   0168    32   016E    33   01A2
35   01AE    36   01B0

ENTRY=0004
PROGRAM SIZE=01B2 BYTES
DATA SIZE=0050 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

```
T' TXDS FORTRAN 936873*B 07/12/79   16:51:13    OPTIONS: S           PAGE   1

      SUBROUTINE CYC3
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
      INTEGER TIMED(2)
      EQUIVALENCE (TIME,TIMED)
C
      DO 10 I=1,10
      DO 10 J=1,2
   10 XS(J,I) = 0
      IF(TIMED(1).GE.2) GO TO 40
      DO 20 I=1,3
      GSQE(I) = 0
   20 GSQL(I) = 0
      DO 30 I=1,2
      DZP(I) = 0.
      GE(I) = 0
   30 GL(I) = 0
      GO TO 50
   40 CONTINUE
C
C     CHANNEL TRANSFER SEQUENCE
C
      GSQL(1) = GSQL(1)*SZP2(JS,JSTEMP)
      GSQL(2) = GSQL(2)*SZP(JS,JSTEMP)
C
      S1 = GSQL(1) + GSQLO(1)
      S2 = GSQL(2)
      S3 = GSQL(3) + GSQLO(2)
C
C     NEW GRAD L
C
      DZP(1) = ZP(JSTEMP,1) - ZP1
      DZP(2) = ZP(JSTEMP,2) - ZP2
C
      GL(1) = S1*DZP(1) + S2*DZP(2)
      GL(2) = S2*DZP(1) + S3*DZP(2)
C
   50 JS = JSTEMP
C
      RETURN
      END
```

COMMON BLOCK/LOGL / ALLOCATION 0008 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	CHC	LOGICAL	2	SCALAR	0002	CHCHAN	LOGICAL	2	SCALAR
0004	EST	LOGICAL	2	SCALAR	0006	MLE	LOGICAL	2	SCALAR

COMMON BLOCK/INT2 / ALLOCATION 046A BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	C1HP	INTEGER*2	2	SCALAR	0002	C2HP	INTEGER*2	2	SCALAR
0004	D	INTEGER*2	160	ARRAY	00A4	DS	INTEGER*2	320	ARRAY
01E4	E1P	INTEGER*2	2	SCALAR	01E6	E2P	INTEGER*2	2	SCALAR
01E8	F	INTEGER*2	130	ARRAY	026A	GK	INTEGER*2	160	ARRAY
030A	JS	INTEGER*2	2	SCALAR	030C	JSTEMP	INTEGER*2	2	SCALAR
030E	MODE	INTEGER*2	2	SCALAR	0310	NC	INTEGER*2	2	SCALAR
0312	NP	INTEGER*2	2	SCALAR	0314	Q	INTEGER*2	176	ARRAY
03C4	SGANS	INTEGER*2	2	SCALAR	03C6	X	INTEGER*2	100	ARRAY
042A	XS	INTEGER*2	40	ARRAY	0452	XY	INTEGER*2	12	ARRAY
045E	Y	INTEGER*2	6	ARRAY	0464	YP	INTEGER*2	6	ARRAY

COMMON BLOCK/INT4 / ALLOCATION 0060 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	ANS	INTEGER*4	4	SCALAR	0004	ANSI	INTEGER*4	4	SCALAR
0008	DT	INTEGER*4	4	SCALAR	000C	GE	INTEGER*4	8	ARRAY
0014	GL	INTEGER*4	8	ARRAY	001C	GSQE	INTEGER*4	12	ARRAY
0028	GSQL	INTEGER*4	12	ARRAY	0034	TJ	INTEGER*4	20	ARRAY
0048	TIME	INTEGER*4	4	SCALAR	004C	TL	INTEGER*4	20	ARRAY
0048	TIMED	INTEGER*2	4	ARRAY					

COMMON BLOCK/REAL / ALLOCATION 0144 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	DZP	REAL	8	ARRAY	0008	GSQLO	REAL	8	ARRAY
0010	RTJC	REAL	4	SCALAR	0014	RTJS	REAL	4	SCALAR
0018	RTJZ	REAL	4	SCALAR	001C	SZP	REAL	100	ARRAY
0080	SZP2	REAL	100	ARRAY	00E4	THRTJC	REAL	4	SCALAR
00E8	THRTJZ	REAL	4	SCALAR	00EC	ZP	REAL	40	ARRAY
0114	ZPS	REAL	20	ARRAY	0128	ZP1	REAL	4	SCALAR
012C	ZP1MAX	REAL	4	SCALAR	0130	ZP1MIN	REAL	4	SCALAR
0134	ZP2	REAL	4	SCALAR	0138	ZP2MAX	REAL	4	SCALAR
013C	ZP2MIN	REAL	4	SCALAR	0140	Z1MIN	REAL	4	SCALAR

```
SCALAR ALLOCATION

LOCN NAME    MODE        BYTES TYPE      LOCN NAME    MODE        BYTES TYPE

0030 I       INTEGER*2       2 SCALAR    0032 J       INTEGER*2       2 SCALAR
0034 S1      REAL            4 SCALAR    0038 S2      REAL            4 SCALAR
003C S3      REAL            4 SCALAR
```

```
SUBPROGRAMS CALLED

NAME    TYPE       ARGS   NAME    TYPE       ARGS   NAME    TYPE       ARGS

F$RREL  RUNTIME           F$RGMY  RUNTIME           F$REL   RUNTIME
F$RITP  RUNTIME


STATEMENT LABELS

LOCN  LABEL USE         LOCN  LABEL USE         LOCN  LABEL USE

001C  10    DO END      00C0  40                006A  20    DO END
00A6  30    DO END      01E4  50                0016  M5
001C  M6                00C0  M7                00C0  M8
0058  M9                0088  M10               01BC  M11
01C0  M12


STATEMENT LOCATIONS

LINE LOCN   LINE LOCN   LINE LOCN   LINE LOCN   LINE LOCN   LINE LOCN

1    0000   3    0010   4    0010   7    0010   9    0010   12   0010
13   0010   14   0010   15   0010   16   0010   18   0010   19   0016
20   001C   21   0048   22   0052   23   0058   24   006A   25   0082
26   0088   27   009A   28   00A6   29   00BE   30   00C0   34   00C0
35   00EC   37   0114   38   0128   39   0146   43   0168   44   0180
46   018C   47   01B0   49   01E4   51   01EA   52   01EC

ENTRY=0004
PROGRAM SIZE=01FC BYTES
DATA SIZE=004E BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

```
TI TXDS FORTRAN 936873*B 07/12/79   16:57:40    OPTIONS: S           PAGE   1

         SUBROUTINE CYC4
 C
         COMMON/LOGL/ CHC,CHCHAN,EST,MLE
         COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
        1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
        2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
         COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
        1             TJ(5),TIME,TL(5)
         COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
        1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
        2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
         LOGICAL CHC,CHCHAN,EST,MLE
         INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
         INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
 C
 C       NEWTON-RAPHSON INCREMENTS
 C
         R1 = GSQL(1) + GSQLO(1)
         R2 = GSQL(2)
         R3 = GSQL(3) + GSQLO(2)
         RGL1 = GL(1)
         RGL2 = GL(2)
 C
         DET = R1*R3 - R2*R2
         IF(DET.EQ.0.) STOP 4
         DETI = 1./DET
 C
         DZP(1) = (R2*RGL2 - R3*RGL1)*DETI
         DZP(2) = (R2*RGL1 - R1*RGL2)*DETI
 C
         RETURN
         END
```

COMMON BLOCK/LOGL / ALLOCATION 0008 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	CHC	LOGICAL	2	SCALAR	0002	CHCHAN	LOGICAL	2	SCALAR
0004	EST	LOGICAL	2	SCALAR	0006	MLE	LOGICAL	2	SCALAR

COMMON BLOCK/INT2 / ALLOCATION 046A BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	C1HP	INTEGER*2	2	SCALAR	0002	C2HP	INTEGER*2	2	SCALAR
0004	D	INTEGER*2	160	ARRAY	00A4	DS	INTEGER*2	320	ARRAY
01E4	E1P	INTEGER*2	2	SCALAR	01E6	E2P	INTEGER*2	2	SCALAR
01E8	F	INTEGER*2	130	ARRAY	026A	GK	INTEGER*2	160	ARRAY
030A	JS	INTEGER*2	2	SCALAR	030C	JSTEMP	INTEGER*2	2	SCALAR
030E	MODE	INTEGER*2	2	SCALAR	0310	NC	INTEGER*2	2	SCALAR
0312	NP	INTEGER*2	2	SCALAR	0314	Q	INTEGER*2	176	ARRAY
03C4	SGANS	INTEGER*2	2	SCALAR	03C6	X	INTEGER*2	100	ARRAY
042A	XS	INTEGER*2	40	ARRAY	0452	XY	INTEGER*2	12	ARRAY
045E	Y	INTEGER*2	6	ARRAY	0464	YP	INTEGER*2	6	ARRAY

COMMON BLOCK/INT4 / ALLOCATION 0060 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	ANS	INTEGER*4	4	SCALAR	0004	ANSI	INTEGER*4	4	SCALAR
0008	DT	INTEGER*4	4	SCALAR	000C	GE	INTEGER*4	8	ARRAY
0014	GL	INTEGER*4	8	ARRAY	001C	GSQE	INTEGER*4	12	ARRAY
0028	GSQL	INTEGER*4	12	ARRAY	0034	TJ	INTEGER*4	20	ARRAY
0048	TIME	INTEGER*4	4	SCALAR	004C	TL	INTEGER*4	20	ARRAY

COMMON BLOCK/REAL / ALLOCATION 0144 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	DZP	REAL	8	ARRAY	0008	GSQLO	REAL	8	ARRAY
0010	RTJC	REAL	4	SCALAR	0014	RTJS	REAL	4	SCALAR
0018	RTJZ	REAL	4	SCALAR	001C	SZP	REAL	100	ARRAY
0080	SZP2	REAL	100	ARRAY	00E4	THRTJC	REAL	4	SCALAR
00E8	THRTJZ	REAL	4	SCALAR	00EC	ZP	REAL	40	ARRAY
0114	ZPS	REAL	20	ARRAY	0128	ZP1	REAL	4	SCALAR
012C	ZP1MAX	REAL	4	SCALAR	0130	ZP1MIN	REAL	4	SCALAR
0134	ZP2	REAL	4	SCALAR	0138	ZP2MAX	REAL	4	SCALAR
013C	ZP2MIN	REAL	4	SCALAR	0140	Z1MIN	REAL	4	SCALAR

SCALAR ALLOCATION

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0030	R1	REAL	4	SCALAR	0034	R2	REAL	4	SCALAR
0038	R3	REAL	4	SCALAR	003C	RGL1	REAL	4	SCALAR

```
0040 RGL2    REAL          4 SCALAR   0044 DET    REAL          4 SCALAR
0048 DETI    REAL          4 SCALAR
```

```
SUBPROGRAMS CALLED

NAME     TYPE         ARGS   NAME     TYPE         ARGS   NAME     TYPE         ARGS

F$RSTO   RUNTIME             F$RREL   RUNTIME             F$RGMY   RUNTIME
F$REL    RUNTIME             F$RITP   RUNTIME


STATEMENT LABELS

LOCN   LABEL USE          LOCN   LABEL USE          LOCN   LABEL USE

00BC   M0                 00BC   M1                 00BC   M2


STATEMENT LOCATIONS

LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN    LINE LOCN

1    0000    3    0010    4    0010    7    0010    9    0010    12   0010
13   0010    14   0010    18   0010    19   0024    20   0042    21   0064
22   0076    24   0094    25   00B0    26   00BC    28   00CC    29   00EE
31   0110    32   0114

ENTRY=0004
PROGRAM SIZE=0118 BYTES
DATA SIZE=005E BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

```
      SUBROUTINE CYC5
C
      COMMON/LOGL/ CHC,CHCHAN,EST,MLE
      COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
      COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1             TJ(5),TIME,TL(5)
      COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
      LOGICAL CHC,CHCHAN,EST,MLE
      INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
      INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
C
C     UPDATE ZP
C
      ZP1 = ZP(JS,1) + DZP(1)
      ZP2 = ZP(JS,2) + DZP(2)
C
      IF(ZP1.GT.ZP1MAX) ZP1 = ZP1MAX
      IF(ZP2.GT.ZP2MAX) ZP2 = ZP2MAX
      IF(ZP1.LT.ZP1MIN) ZP1 = ZP1MIN
      IF(ZP2.LT.ZP1MIN) ZP2 = ZP2MIN
C
      RETURN
      END
```

COMMON BLOCK/LOGL / ALLOCATION 0008 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	CHC	LOGICAL	2	SCALAR	0002	CHCHAN	LOGICAL	2	SCALAR
0004	EST	LOGICAL	2	SCALAR	0006	MLE	LOGICAL	2	SCALAR

COMMON BLOCK/INT2 / ALLOCATION 046A BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	C1HP	INTEGER*2	2	SCALAR	0002	C2HP	INTEGER*2	2	SCALAR
0004	D	INTEGER*2	160	ARRAY	00A4	DS	INTEGER*2	320	ARRAY
01E4	E1P	INTEGER*2	2	SCALAR	01E6	E2P	INTEGER*2	2	SCALAR
01E8	F	INTEGER*2	130	ARRAY	026A	GK	INTEGER*2	160	ARRAY
030A	JS	INTEGER*2	2	SCALAR	030C	JSTEMP	INTEGER*2	2	SCALAR
030E	MODE	INTEGER*2	2	SCALAR	0310	NC	INTEGER*2	2	SCALAR
0312	NP	INTEGER*2	2	SCALAR	0314	Q	INTEGER*2	176	ARRAY
03C4	SGANS	INTEGER*2	2	SCALAR	03C6	X	INTEGER*2	100	ARRAY
042A	XS	INTEGER*2	40	ARRAY	0452	XY	INTEGER*2	12	ARRAY
045E	Y	INTEGER*2	6	ARRAY	0464	YP	INTEGER*2	6	ARRAY

COMMON BLOCK/INT4 / ALLOCATION 0060 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	ANS	INTEGER*4	4	SCALAR	0004	ANSI	INTEGER*4	4	SCALAR
0008	DT	INTEGER*4	4	SCALAR	000C	GE	INTEGER*4	8	ARRAY
0014	GL	INTEGER*4	8	ARRAY	001C	GSQE	INTEGER*4	12	ARRAY
0028	GSQL	INTEGER*4	12	ARRAY	0034	TJ	INTEGER*4	20	ARRAY
0048	TIME	INTEGER*4	4	SCALAR	004C	TL	INTEGER*4	20	ARRAY

COMMON BLOCK/REAL / ALLOCATION 0144 BYTES

LOCN	NAME	MODE	BYTES	TYPE	LOCN	NAME	MODE	BYTES	TYPE
0000	DZP	REAL	8	ARRAY	0008	GSQLO	REAL	8	ARRAY
0010	RTJC	REAL	4	SCALAR	0014	RTJS	REAL	4	SCALAR
0018	RTJZ	REAL	4	SCALAR	001C	SZP	REAL	100	ARRAY
0080	SZP2	REAL	100	ARRAY	00E4	THRTJC	REAL	4	SCALAR
00E8	THRTJZ	REAL	4	SCALAR	00EC	ZP	REAL	40	ARRAY
0114	ZPS	REAL	20	ARRAY	0128	ZP1	REAL	4	SCALAR
012C	ZP1MAX	REAL	4	SCALAR	0130	ZP1MIN	REAL	4	SCALAR
0134	ZP2	REAL	4	SCALAR	0138	ZP2MAX	REAL	4	SCALAR
013C	ZP2MIN	REAL	4	SCALAR	0140	Z1MIN	REAL	4	SCALAR

```
SUBPROGRAMS CALLED

NAME     TYPE       ARGS   NAME     TYPE       ARGS   NAME     TYPE       ARGS

F$RREL   RUNTIME           F$RGMY   RUNTIME           F$RITP   RUNTIME

STATEMENT LABELS

LOCN  LABEL USE           LOCN  LABEL USE           LOCN  LABEL USE

004E  M0                  006E  M1                  008E  M2
00AE  M3                  004E  M4                  004E  M5
006E  M6                  006E  M7                  008E  M8
008E  M9                  00AE  M10                 00AE  M11

STATEMENT LOCATIONS

LINE LOCN   LINE LOCN   LINE LOCN   LINE LOCN   LINE LOCN   LINE LOCN

1    0000   3    0010   4    0010   7    0010   9    0010   12   0010
13   0010   14   0010   18   0010   19   0026   21   0032   22   004E
23   006E   24   008E   26   00AE   27   00B0

ENTRY=0004
PROGRAM SIZE=00B0 BYTES
DATA SIZE=0030 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS
```

APPENDIX D

SCALE-FORTRAN LISTING

7751T 01   06-27-79   16.769   PAGE   1

```
      COMMON/VARDAT/UX(50),LX(50)
C
      COMMON/DAT/X(5,10),TJ(5),TL(5),F(5,17),D(5,16),E1P,E2P,SIGSQ,ANS
C
      COMMON/SENSP/XS(4,10),DS(5,4,16),GE(4),GL(4),
     1          GSQE(10),GSQL(10),GSQLO(4),GK(5,4,8)
C
      COMMON/MEAS/Y(3),YP(3),XY(2,3),DT,TIME,MODE
C
      COMMON/PARM/ZP(5,4),DZP(4),ZP1,ZP2,ZP3,ZP4,Z1MIN,
     1                NP,NC,JSTEMP,JS,CHCHAN
C
      COMMON/TSW/RTJS,THRTJC,RTJC,THRTJZ,RTJZ
C
      COMMON/PLIM/ZP1MAX,ZP2MAX,ZP3MAX,ZP4MAX,ZP1MIN,ZP2MIN,ZP3MIN,
     1  ZP4MIN
C
      COMMON/FHDAT/DHP,WHP,C1HP,C2HP
C
      COMMON/CSCALE/SCALE(60)
C
      INTEGER CHCHAN
      INTEGER DSCL(16),DSSCL(16),FSCL(13),GKSCL(8),
     1          XSCL(10),XSSCL(10),YPSCL(3),Q(88),T1,T2,S1,S11,S12
      DIMENSION SZP(5,5),SZP2(5,5),ZPS(5)
C
      READ(5,1000) XSCL,XSSCL,YPSCL
 1000 FORMAT(2(10I4/),3I4)
      READ(2) MODE,NP,NC,JSTEMP,JS,CHCHAN,
     1          C1HP,C2HP,DT,TIME,X,XS,XY,Y,YP,D,DS,F,GK,
     2          TJ,E1P,E2P,SIGSQ,ANS,GE,GL,GSQE,GSQL,GSQLO,
     3          ZP,DZP,ZP1,ZP2,ZP3,ZP4,Z1MIN,
     4          RTJS,THRTJC,RTJC,THRTJZ,RTJZ,
     5          ZP1MAX,ZP2MAX,ZP3MAX,ZP4MAX,
     6          ZP1MIN,ZP2MIN,ZP3MIN,ZP4MIN
C
      SCALE(21) = 1.
      DO 10 I=1,20
   10 SCALE(21-I) = SCALE(22-I)/2.
      DO 20 I=22,60
   20 SCALE(I) = SCALE(I-1)*2.
      DT = DT*SCALE(37)
      DO 70 K=1,16
      DSCL(K) = ISCALE(D(1,K),5)
      SCLD = SCALE(36-DSCL(K))
      DSSCL(K) = ISCALE(DS(1,1,K),10)
      SCLDS = SCALE(36-DSSCL(K))
      IF(K.GT.13) GO TO 30
      FSCL(K) = ISCALE(F(1,K),5)
      SCLF = SCALE(36-FSCL(K))
      IF(K.GT.8) GO TO 30
      GKSCL(K) = ISCALE(GK(1,1,K),10)
      SCLGK = SCALE(36-GKSCL(K))
```

```
7751T 01   06-27-79   16.771                                                    PAGE     1

      FUNCTION ISCALE(A,N)
      COMMON/CSCALE/SCALE(60)
      DIMENSION A(1)
      ISCL = 1
      SCL = SCALE(ISCL)
      DO 10 I=1,N
      ABSA = ABS(A(I))
    5 CONTINUE
      IF(ABSA.LT.SCL) GO TO 10
      ISCL = ISCL+1
      SCL = SCALE(ISCL)
      GO TO 5
   10 CONTINUE
      ISCALE = ISCL-21
      RETURN
      END
```

HONEYWELL PAGE PRINTING SYSTEM- #1168-02

```
      30 DO 60 I=1,5                                          00000590
         D(I,K) = D(I,K)*SCLD                                 00000600
         DO 40 J=1,NP                                         00000610
      40 DS(I,J,K) = DS(I,J,K)*SCLDS                          00000620
         IF(K.GT.13) GO TO 60                                 00000630
         F(I,K) = F(I,K)*SCLF                                 00000640
         IF(K.GT.8) GO TO 60                                  00000650
         DO 50 J=1,NP                                         00000660
      50 GK(I,J,K) = GK(I,J,K)*SCLGK                          00000670
      60 CONTINUE                                             00000680
      70 CONTINUE                                             00000690
         Q(64) = 1-XSCL(10)+DSCL(14)+XSCL(1)                  00000700
         Q(2) = 1-XSCL(10)+DSCL(15)+XSCL(2)                   00000710
         Q(3) = 1-XSCL(10)+DSCL(16)+XSCL(4)                   00000720
         Q(4) = 1-XSCL(5)+DSCL(1)+XSCL(1)                     00000730
         Q(5) = 1-XSCL(5)+DSCL(2)+XSCL(2)                     00000740
         Q(6) = 1-XSCL(5)+DSCL(3)+XSCL(3)                     00000750
         Q(7) = 1-XSCL(5)+DSCL(4)+XSCL(4)                     00000760
         Q(8) = 1-XSCL(5)+DSCL(11)+YPSCL(3)                   00000770
         Q(9) = 1-XSCL(5)+FSCL(6)+XSCL(9)                     00000780
         Q(10) = 1-XSCL(5)+FSCL(10)+XSCL(10)                  00000790
         Q(11) = 1-XSCL(6)+DSCL(5)+XSCL(1)                    00000800
         Q(12) = 1-XSCL(6)+DSCL(6)+XSCL(2)                    00000810
         Q(13) = 1-XSCL(6)+DSCL(7)+XSCL(3)                    00000820
         Q(14) = 1-XSCL(6)+DSCL(8)+XSCL(4)                    00000830
         Q(15) = 1-XSCL(6)+DSCL(12)+YPSCL(3)                  00000840
         Q(16) = 1-XSCL(6)+FSCL(7)+XSCL(9)                    00000850
         Q(17) = 1-XSCL(6)+FSCL(11)+XSCL(10)                  00000860
         Q(18) = 1-XSCL(7)+DSCL(9)+XSCL(3)                    00000870
         Q(19) = 1-XSCL(7)+FSCL(8)+XSCL(9)                    00000880
         Q(20) = 1-XSCL(7)+FSCL(12)+XSCL(10)                  00000890
         Q(21) = 1-XSCL(8)+DSCL(10)+XSCL(4)                   00000900
         Q(22) = 1-XSCL(8)+DSCL(13)+YPSCL(3)                  00000910
         Q(23) = 1-XSCL(8)+FSCL(9)+XSCL(9)                    00000920
         Q(24) = 1-XSCL(8)+FSCL(13)+XSCL(10)                  00000930
         Q(25) = 0                                            00000940
         Q(26) = -30+FSCL(1)+2*XSCL(9)                        00000950
         Q(27) = -30+FSCL(4)+2*XSCL(10)-Q(26)                 00000960
         IF(Q(27).LE.0) GO TO 80                              00000970
         Q(25) = -Q(27)                                       00000980
         Q(26) = Q(27)+Q(26)                                  00000990
         Q(27) = 0                                            00001000
      80 Q(26) = -29+FSCL(2)+XSCL(9)+XSCL(10)-Q(26)           00001010
         Q(29) = 1-XSSCL(10)+DSCL(14)+XSSCL(1)                00001020
         Q(30) = 1-XSSCL(10)+DSCL(15)+XSSCL(2)                00001030
         Q(31) = 1-XSSCL(10)+DSCL(16)+XSSCL(4)                00001040
         Q(32) = 1-XSSCL(10)+DSSCL(14)+XSCL(1)                00001050
         Q(33) = 1-XSSCL(10)+DSSCL(15)+XSCL(2)                00001060
         Q(34) = 1-XSSCL(10)+DSSCL(16)+XSCL(4)                00001070
         Q(35) = 1-XSSCL(5)+DSCL(1)+XSSCL(1)                  00001080
         Q(36) = 1-XSSCL(5)+DSCL(2)+XSSCL(2)                  00001090
         Q(37) = 1-XSSCL(5)+DSCL(3)+XSSCL(3)                  00001100
         Q(38) = 1-XSSCL(5)+DSCL(4)+XSSCL(4)                  00001110
```

```
Q(39) = 1-XSSCL(5)+DSSCL(1)+XSCL(1)
Q(40) = 1-XSSCL(5)+DSSCL(2)+XSCL(2)
Q(41) = 1-XSSCL(5)+DSSCL(3)+XSCL(3)
Q(42) = 1-XSSCL(5)+DSSCL(4)+XSCL(4)
Q(43) = 1-XSSCL(5)+DSSCL(11)+YPSCL(3)
Q(44) = 1-XSSCL(5)+FSCL(6)+XSSCL(9)
Q(45) = 1-XSSCL(5)+FSCL(10)+XSSCL(10)
Q(46) = 1-XSSCL(5)+GKSCL(1)+XSCL(9)
Q(47) = 1-XSSCL(5)+GKSCL(5)+XSCL(10)
Q(48) = 1-XSSCL(6)+DSCL(5)+XSSCL(1)
Q(49) = 1-XSSCL(6)+DSCL(6)+XSSCL(2)
Q(50) = 1-XSSCL(6)+DSCL(7)+XSSCL(3)
Q(51) = 1-XSSCL(6)+DSCL(8)+XSSCL(4)
Q(52) = 1-XSSCL(6)+DSSCL(5)+XSCL(1)
Q(53) = 1-XSSCL(6)+DSSCL(6)+XSCL(2)
Q(54) = 1-XSSCL(6)+DSSCL(7)+XSCL(3)
Q(55) = 1-XSSCL(6)+DSSCL(8)+XSCL(4)
Q(56) = 1-XSSCL(6)+DSSCL(12)+YPSCL(3)
Q(57) = 1-XSSCL(6)+FSCL(7)+XSSCL(9)
Q(58) = 1-XSSCL(6)+FSCL(11)+XSSCL(10)
Q(59) = 1-XSSCL(6)+GKSCL(2)+XSCL(9)
Q(60) = 1-XSSCL(6)+GKSCL(6)+XSCL(10)
Q(61) = 1-XSSCL(7)+DSCL(9)+XSSCL(3)
Q(62) = 1-XSSCL(7)+FSCL(8)+XSSCL(9)
Q(63) = 1-XSSCL(7)+FSCL(12)+XSSCL(10)
Q(64) = 1-XSSCL(7)+GKSCL(3)+XSCL(9)
Q(65) = 1-XSSCL(7)+GKSCL(7)+XSCL(10)
Q(66) = 1-XSSCL(8)+DSCL(10)+XSSCL(4)
Q(67) = 1-XSSCL(8)+FSCL(9)+XSSCL(9)
Q(68) = 1-XSSCL(8)+FSCL(13)+XSSCL(10)
Q(69) = 1-XSSCL(8)+GKSCL(4)+XSCL(9)
Q(70) = 1-XSSCL(8)+GKSCL(8)+XSCL(10)
T1 = XSSCL(9)+FSCL(1)
Q(71) = -T1+FSCL(2)+XSSCL(10)
T2 = XSSCL(10)+FSCL(4)
Q(72) = -T2+FSCL(3)+XSSCL(9)
S11 = T1+XSCL(9)
S12 = T2+XSCL(10)
S1 = S11
IF(S1.LT.S12) S1 = S12
Q(73) = S11-S1
Q(74) = S12-S1
ISCLA = ISCALE(1./(1.-EIP))
Q(75) = -ISCLA
Q(85) = -S1+S1+ISCLA
S11 = T1+XSSCL(9)
S12 = T2+XSSCL(10)
S1 = S11
IF(S1.LT.S12) S1 = S12
Q(76) = S11-S1
Q(77) = S12-S1
Q(78) = -S1+S1-Q(85)
Q(79) = ISCALE(C1HP,1)+1
```

7751T 01 06-27-79   16.769   PAGE   4

```
           C1HP = C1HP*SCALE(37-Q(79))                                               00001650
           Q(80) = ISCALE(C2HP,1)+1                                                  00001660
           C2HP = C2HP*SCALE(37-Q(80))                                               00001670
           SGANS = 2.*(E1P*ANS+1.)                                                   00001680
           ISCLSA = ISCLA + 2                                                        00001690
           ANS1 = SCALE(52-ISCLA)                                                    00001700
           ANS = ANS1                                                                00001710
           E1P = (E1P-1.)*SCALE(37)                                                  00001720
           E2P = (E2P-1.)*SCALE(37)                                                  00001730
           SGANS = SGANS*SCALE(36-ISCLSA)                                            00001740
           Q(82) = ISCLSA-15                                                         00001750
           Q(83) = -Q(82)                                                            00001760
           Q(81) = Q(82)+FSCL(5)                                                     00001770
           Q(87) = 0                                                                 00001780
           Q(88) = 1                                                                 00001790
           GSQLO(1) = GSQLO(1)*SCALE(21-Q(85))                                       00001800
           GSQLO(2) = GSQLO(2)*SCALE(21-Q(85))                                       00001810
           DO 90 I=1,5                                                               00001820
           ZPS(I) = ZP(I,1)**2/SCALE(21-Q(85))                                       00001830
           DO 90 J=1,5                                                               00001840
           SZP(I,J) = 1.+ZP(I,1)/ZP(J,1)                                             00001850
        90 SZP2(I,J) = SZP(I,J)**2                                                   00001860
       200 CALL FMEDIA(3,5)                                                          00001870
           WRITE(3,2000) JS,JSTEMP,MODE,Q,                                           00001880
          1                C1HP,C2HP,D,(((DS(I,J,K),I=1,5),J=1,2),K=1,16),           00001890
          2                E1P,E2P,((F(I,J),I=1,5),J=1,13),                          00001900
          3                (((GK(I,J,K),I=1,5),J=1,2),K=1,8),SGANS,X,                00001910
          4                ((XS(I,J),I=1,2),J=1,10),XY,Y,YP,ANS,ANS1,DT,             00001920
          5                GE(1),GE(2),GL(1),GL(2),(GSQE(I),I=1,3),                  00001930
          6                (GSQL(I),I=1,5),TJ,TIME,TL                                00001940
           WRITE(3,3000) DZP(1),DZP(2),(GSQLO(I),I=1,2),RTJC,RTJS,RTJZ,              00001950
          1                SZP,SZP2,THRTJC,THRTJZ,((ZP(I,J),I=1,5),J=1,2),           00001960
          2                ZPS,ZP1,ZP1MAX,ZP1MIN,ZP2,ZP2MAX,ZP2MIN,Z1MIN             00001970
      2000 FORMAT(3I9,11(/8I9),62(/8F9.0))                                           00001980
      3000 FORMAT(4E18.8)                                                            00001990
     C                                                                               00002000
           WRITE(9,4000) DSCL,DSSCL,FSCL,GKSCL,XSCL,XSSCL,YPSCL,Q                    00002010
      4000 FORMAT(" D  SCALE",16I4/" DS SCALE",16I4/" F  SCALE",13I4/                00002020
          1       " GK SCALE",8I4/" X  SCALE",10I4/" XS SCALE",10I4/                 00002030
          2       " YP SCALE",3I4/" Q  SCALE",10I4/8(9X,10I4/)//)                    00002040
           STOP                                                                      00002050
           END                                                                       00002060
```

HONEYWELL PAGE PRINTING SYSTEM—P1185-02

APPENDIX E

FUNDAMENTAL OPERATION SUBPROGRAMS (FOS): TI990 ASSEMBLY LISTINGS

```
MD              TXMIRA        2.3.0  78.244 09:36:23    07/10/79     PAGE 0001

                             IDT    'MD'
                    *
                    *   FUNCTION MD(I,X)
                    *   INTEGER I
                    *   INTEGER*4 X,MD
                    *   MULTIPLY I BY X AND RETURN THE
                    *   TWO HIGHEST ORDER WORDS
                    *
                             DEF    MD
     0000                    DSEG
     0000           $DATA    BSS    32
     0020                    DEND
     0000           MD       PSEG
     0000 0000"              DATA   $DATA
     0002 0004'              DATA   MD+4
     0004                    RORG   4
     0004 05CE               INCT    14
     0006 C07E               MOV    *14+,1        (1)= ADDRESS OF I
     0008 C0BE               MOV    *14+,2        (2)+(3)= ADDRESSES OF X
     000A C0C2               MOV    2,3
     000C 05C3               INCT   3
     000E C192               MOV    *2,6
     0010 110A               JLT    $L2
                    * X>=0
     0012 C113               MOV    *3,4
     0014 3911               MPY    *1,4
     0016 3991               MPY    *1,6
     0018 A1C4               A      4,7
     001A 1701               JNC    $L1
     001C 0586               INC    6
     001E C04D      $L1      MOV    13,1
     0020 CC46               MOV    6,*1+
     0022 C447               MOV    7,*1
     0024 0380               RTWP
                    * X<0
     0026 C113      $L2      MOV    *3,4
     0028 0504               NEG    4
     002A 1601               JNE    $L3
     002C 0606               DEC    6
     002E 0546      $L3      INV    6
     0030 3911               MPY    *1,4
     0032 3991               MPY    *1,6
     0034 A1C4               A      4,7
     0036 1701               JNC    $L4
     0038 0586               INC    6
     003A 0507      $L4      NEG    7
     003C 1601               JNE    $L5
     003E 0606               DEC    6
     0040 0546      $L5      INV    6
     0042 C04D               MOV    13,1
     0044 CC46               MOV    6,*1+
     0046 C447               MOV    7,*1
     0048 0380               RTWP
     004A                    PEND
```

```
 0055                     END
```

```
 " $DATA    0000    ' $L1     001E    ' $L2     0026    ' $L3     002E
 ' $L4      003A    ' $L5     0040    D MD      0000

0000 ERRORS
```

```
                     IDT   'MS'
               *
               *   FUNCTION MS(I,J,Q)
               *   INTEGER*4 MS
               *   MULTIPLY I BY J AND SHIFT THE 2-WORD RESULT
               *   BY Q BITS
               *
                     DEF   MS
0000                 DSEG
0000           $DATA BSS   32
0020                 DEND
0000           MS    PSEG
0000 0000"           DATA  $DATA
0002 0004'           DATA  MS+4
0004                 RORG  4
               *
0004 05CE            INCT  14
0006 C13E            MOV   *14+,4      (4)= ADDRESS OF I
0008 C17E            MOV   *14+,5      (5)= ADDRESS OF J
000A C2BE            MOV   *14+,10
000C C19A            MOV   *10,6       (6)= ADDRESS OF Q
000E C054            MOV   *4,1
0010 1104            JLT   $L1
0012 C095            MOV   *5,2
0014 110B            JLT   $L3
               * I>=0, J>=0
0016 3881            MPY   1,2
0018 1013            JMP   SHFT
001A C095      $L1   MOV   *5,2
001C 110E            JLT   $L5
               * I<0, J>=0
001E 0501            NEG   1
0020 3881            MPY   1,2
0022 0503            NEG   3
0024 1601            JNE   $L2
0026 0602            DEC   2
0028 0542      $L2   INV   2
002A 100A            JMP   SHFT
               * I>=0, J<0
002C 0502      $L3   NEG   2
002E 3881            MPY   1,2
0030 0503            NEG   3
0032 1601            JNE   $L4
0034 0602            DEC   2
0036 0542      $L4   INV   2
0038 1003            JMP   SHFT
               * I<0, J<0
003A 0501      $L5   NEG   1
003C 0502            NEG   2
003E 3881            MPY   1,2
               *
               *  THIS PART IS SIMILAR TO SD
               *
0040 0208      SHFT  LI    8,16
```

```
MS              TXMIRA        2.3.0  78.244 11:55:33    07/13/79     PAGE 0002

        0042 0010
   0055 0044 C016          MOV    *6,0
   0056 0046 1112          JLT    $L6
   0057 0048 130D          JEQ    SHFTO
   0058 004A 6200          S      0,8
   0059 004C 0A02          SLA    2,0
   0060 004E C240          MOV    0,9
   0061 0050 0A19          SLA    9,1
   0062 0052 C1A9          MOV    @WRD(9),6
        0054 0020"
   0063 0056 C008          MOV    8,0
   0064 0058 0B03          SRC    3,0
   0065 005A C143          MOV    3,5
   0066 005C 4146          SZC    6,5
   0067 005E 0546          INV    6
   0068 0060 40C6          SZC    6,3
   0069 0062 E085          SOC    5,2
   0070 0064 C04D  SHFTO   MOV    13,1
   0071 0066 CC42          MOV    2,*1+
   0072 0068 C443          MOV    3,*1
   0073 006A 0380          RTWP
   0074 006C 0740  $L6     ABS    0
   0075 006E 0903          SRL    3,0
   0076 0070 C240          MOV    0,9
   0077 0072 0A19          SLA    9,1
   0078 0074 C1A9          MOV    @WRD(9),6
        0076 0020"
   0079 0078 C1C2          MOV    2,7
   0080 007A 41C6          SZC    6,7
   0081 007C 0802          SRA    2,0
   0082 007E 0B07          SRC    7,0
   0083 0080 E0C7          SOC    7,3
   0084 0082 C04D          MOV    13,1
   0085 0084 CC42          MOV    2,*1+
   0086 0086 C443          MOV    3,*1
   0087 0088 0380          RTWP
   0088 008A               PEND
   0089 0020               DSEG
   0090 0020 FFFF  WRD     DATA   >FFFF
   0091 0022 FFFE          DATA   >FFFE
   0092 0024 FFFC          DATA   >FFFC
   0093 0026 FFF8          DATA   >FFF8
   0094 0028 FFF0          DATA   >FFF0
   0095 002A FFE0          DATA   >FFE0
   0096 002C FFC0          DATA   >FFC0
   0097 002E FF80          DATA   >FF80
   0098 0030 FF00          DATA   >FF00
   0099 0032 FE00          DATA   >FE00
   0100 0034 FC00          DATA   >FC00
   0101 0036 F800          DATA   >F800
   0102 0038 F000          DATA   >F000
   0103 003A E000          DATA   >E000
   0104 003C C000          DATA   >C000
   0105 003E 8000          DATA   >8000
```

```
0106 0040 0000          DATA  0
0107 0042               DEND
0108                    END
```

```
 " $DATA   0000    ' $L1     001A    ' $L2     0028    ' $L3     002C
 ' $L4     0036    ' $L5     003A    ' $L6     006C    D MS      0000
 ' SHFT    0040    ' SHFTO   0064    " WRD     0020

 0000 ERRORS
```

```
S              TXMIRA        2.3.0  78.244 09:33:32     07/10/79      PAGE 0001

                         IDT    'S'
                  *
                  *    FUNCTION S(I,J,Q)
                  *    INTEGER I,J,Q,S
                  *    MULTIPLY I BY J AND SHIFT THE FIRST WORD
                  *    OF THE RESULT BY Q BITS
                  *
                         DEF    S
     0000                DSEG
     0000         $DATA  BSS    32
     0020                DEND
     0000         S      PSEG
     0000 0000"          DATA   $DATA
     0002 0004'          DATA   S+4
     0004                RORG   4
                  *
     0004 05CE           INCT   14
     0006 C13E           MOV    *14+,4
     0008 C17E           MOV    *14+,5
     000A C054           MOV    *4,1          (1)= ADDRESS OF I
     000C C095           MOV    *5,2          (2)= ADDRESS OF J
     000E C2BE           MOV    *14+,10
     0010 C0DA           MOV    *10,3         (3)= ADDRESS OF Q
     0012 C111           MOV    *1,4
     0014 1104           JLT    $L1
     0016 C152           MOV    *2,5
     0018 1108           JLT    $L2
                  * I>=0, J>=0
     001A 3944           MPY    4,5
     001C 100D           JMP    SHFT
     001E C152    $L1    MOV    *2,5
     0020 1108           JLT    $L3
                  * I<0, J>=0
     0022 0504           NEG    4
     0024 3944           MPY    4,5
     0026 0505           NEG    5
     0028 1007           JMP    SHFT
                  * I>=0, J<0
     002A 0505    $L2    NEG    5
     002C 3944           MPY    4,5
     002E 0505           NEG    5
     0030 1003           JMP    SHFT
                  * I<0, J<0
     0032 0504    $L3    NEG    4
     0034 0505           NEG    5
     0036 3944           MPY    4,5
     0038 C013    SHFT   MOV    *3,0
     003A 1104           JLT    $L4
     003C 1305           JEQ    SHFT0
     003E 0A05           SLA    5,0
     0040 C745           MOV    5,*13
     0042 0380           RTWP
     0044 0500    $L4    NEG    0
     0046 0805           SRA    5,0
```

```
0055 0048 C745  SHFTO    MOV    5,*13
0056 004A 0380           RTWP
0057 004C                PEND
0058                     END
```

```
S               TXMIRA          2.3.0  78.244 09:33:32     07/10/79      PAGE 0003

   " $DATA    0000    ' $L1      001E    ' $L2      002A    ' $L3      0032
   ' $L4      0044    D S        0000    ' SHFT     0038    ' SHFTO    0048

 0000 ERRORS
```

```
SD              TXMIRA        2.3.0  78.244 11:56:42     07/13/79      PAGE 0001

                                IDT    'SD'
                     *
                     *   FUNCTION SD(X,Q)
                     *   INTEGER*4 SD,X
                     *   SHIFT X BY Q BITS
                     *
                                DEF    SD
     0000                       DSEG
     0000            $DATA      BSS    32
     0020                       DEND
     0000            SD         PSEG
     0000 0000"                 DATA   $DATA
     0002 0004'                 DATA   SD+4
     0004                       RORG   4
                     *
     0004 05CE                  INCT   14
     0006 0208                  LI     8,16
     0008 0010
     000A C0BE                  MOV    *14+,2       (2)+(3)= ADDRESSES OF X
     000C C0C2                  MOV    2,3
     000E 05C3                  INCT   3
     0010 C2BE                  MOV    *14+,10
     0012 C15A                  MOV    *10,5        (5)= ADDRESS OF Q
     0014 C015                  MOV    *5,0
     0016 1118                  JLT    $L1
     0018 1313                  JEQ    SHFT0
     001A 6200                  S      0,8
     001C C112                  MOV    *2,4
     001E C153                  MOV    *3,5
     0020 0A04                  SLA    4,0
     0022 C240                  MOV    0,9
     0024 0A19                  SLA    9,1
     0026 C1A9                  MOV    @WRD(9),6
     0028 0020"
     002A C008                  MOV    8,0
     002C 0B05                  SRC    5,0
     002E C0C5                  MOV    5,3
     0030 40C6                  SZC    6,3
     0032 0546                  INV    6
     0034 4146                  SZC    6,5
     0036 E103                  SOC    3,4
     0038 C04D                  MOV    13,1
     003A CC44                  MOV    4,*1+
     003C C445                  MOV    5,*1
     003E 0380                  RTWP
     0040 C04D       SHFT0      MOV    13,1
     0042 CC52                  MOV    *2,*1+
     0044 C453                  MOV    *3,*1
     0046 0380                  RTWP
     0048 0740       $L1        ABS    0
     004A C112                  MOV    *2,4
     004C C153                  MOV    *3,5
     004E 0905                  SRL    5,0
     0050 C240                  MOV    0,9
```

```
 0053 0052 0A19            SLA   9,1
 0054 0054 C1A9            MOV   @WRD(9),6
      0056 0020"
 0055 0058 C1C4            MOV   4,7
 0056 005A 41C6            SZC   6,7
 0057 005C 0804            SRA   4,0
 0058 005E 0B07            SRC   7,0
 0059 0060 E147            SOC   7,5
 0060 0062 C04D            MOV   13,1
 0061 0064 CC44            MOV   4,*1+
 0062 0066 C445            MOV   5,*1
 0063 0068 0380            RTWP
 0064 006A                 PEND
 0065 0020                 DSEG
 0066 0020 FFFF  WRD       DATA  >FFFF
 0067 0022 FFFE            DATA  >FFFE
 0068 0024 FFFC            DATA  >FFFC
 0069 0026 FFF8            DATA  >FFF8
 0070 0028 FFF0            DATA  >FFF0
 0071 002A FFE0            DATA  >FFE0
 0072 002C FFC0            DATA  >FFC0
 0073 002E FF80            DATA  >FF80
 0074 0030 FF00            DATA  >FF00
 0075 0032 FE00            DATA  >FE00
 0076 0034 FC00            DATA  >FC00
 0077 0036 F800            DATA  >F800
 0078 0038 F000            DATA  >F000
 0079 003A E000            DATA  >E000
 0080 003C C000            DATA  >C000
 0081 003E 8000            DATA  >8000
 0082 0040 0000            DATA  >0000
 0083 0042                 DEND
 0084                      END
```

```
 " $DATA    0000     ' $L1       0048    D SD       0000     ' SHFTO    0040
 " WRD      0020
```

0000 ERRORS

APPENDIX F

I/O AND TIMING: TI990 ASSEMBLY LISTING

```
                          IDT   'CLOCK5'          MAY 15,1979
                   *
                   *  THIS MODULE CONTAINS 3 FORTRAN CALLABLE SUBROUTINES
                   *  AND ONE INTERRUPT SERVICE ROUTINE WHICH PROVIDE
                   *  TIMING CONTROL OF REAL TIME OPERATIONS.
                   *     SUBROUTINE TIMEON(N)   STARTS THE CLOCK AND
                   *  PERFORMS TIMING INITIALIZATION. N = NO. OF 8.33 MSEC.
                   *  PULSES PER BASIC PROGRAM CYCLE. IT SHOULD BE CALLED
                   *  ONCE BY THE MAIN PROGRAM.
                   *     SUBROUTINE WAIT       DELAYS FURTHER PROCESSING UNTIL
                   *  THE BASIC SAMPLE TIME HAS EXPIRED(N*8.33). IT SHOULD BE
                   *  CALLED IN THE REAL TIME LOOP AFTER OTHER ROUTINES ARE
                   *  PROCESSED.  -- NO ARGUMENTS --
                   *     SUBROUTINE TIMEOF(MAXCNT,NERR)  STOPS THE CLOCK AND
                   *  PROVIDES AN INDICATION OF MAXIMUM LOOP TIME AND TIMING
                   *  ERRORS THAT OCCURRED DURING THE REAL TIME LOOP. MAXCNT =
                   *  MAX. NO. OF CLOCK PULSES REQUIRED BY RUNTIME ROUTINES.
                   *  NERR = NO. OF PROCESSING TIME OVERFLOWS THAT OCCURRED
                   *  DURING THE REAL TIME LOOP.
                   *     ROUTINE CLK5       SERVICES LEVEL 5 INTERRUPTS
                   *  GENERATED (EVERY 8.33 MSEC.(120 HZ)) BY A REAL TIME
                   *  CLOCK. ITS OPERATION IS AUTOMATICALLY CONTROLLED.
                   *
                          DEF   TIMEON
                          DEF   WAIT
                          DEF   TIMEOF
                   *
                   *
                   *  SUBROUTINE TIMEON
                   *  INITIALIZE CLOCK INTERRUPT PROCESSING AND START THE CLOCK
                   *
0000 0004'         TIMEON DATA  WSP1                SUB. TIMEON WORKSPACE ADDR.
0002 0024'                DATA  PC1                 SUB. TIMEON PC ADDR.
0004               WSP1   BSS   32                  WORKSP. FOR TIMEON,WAIT,TIMEOF
                   *
0024 0300          PC1    LIMI  0                   DISABLE INTERRUPTS
0026 0000
0028 0202                 LI    2,WSP               R2=WP ADDR OF INT. SERV. ROUT.
002A 008E'
002C 0203                 LI    3,CLK5              R3=PC ADDR OF INT. SERV. ROUT.
002E 00AE'
0030 C13E                 MOV   *14+,4              R4=NO. OF SUBROUTINE ARGUMENTS
0032 C13E                 MOV   *14+,4              R4=ADDR. OF 1ST ARGUMENT
                   *                                R14 = CORRECT PC ADDR FOR RTWP
0034 C054                 MOV   *4,1                R1=N=NO. OF PULSES PER CYCLE
0036 C802                 MOV   2,@>14              LOAD LEVEL 5 INT VECT WP
0038 0014
003A C803                 MOV   3,@>16              LOAD LEVEL 5 INT VECT  PC
003C 0016
003E 04C5                 CLR   5
0040 04C6                 CLR   6
0042 026F                 ORI   15,>000F            ENABLE INTERRUPT LEVEL 5 IN
0044 000F
0046 024F                 ANDI  15,>FFF5            STATUS REG OF MAIN PROG.
```

```
0048 FFF5
004A 03A0          CKON                    START THE CLOCK
004C 0300          LIMI 5                  ENABLE LEVEL 5 IN SUB. TIMEON
004E 0005
0050 0340          IDLE                    WAIT FOR 1ST INTERRUPT AND
             *                             SYNCHRONIZE COUNTER IN
0052 04D2          CLR  *2                 REG. 0 OF INT. SERV. ROUT.
0054 0380          RTWP                    RETURN
             *
             *  SUBROUTINE WAIT
             *  DELAY FURTHER PROCESSING UNTIL SAMPLE TIME HAS
             *  EXPIRED - CHECK FOR TIMING ERRORS.
             *
0056 0004' WAIT    DATA WSP1               SUB. WAIT WORKSPACE ADDR.
0058 005A'         DATA PC2                SUB. WAIT PC ADDR.
005A 05CE  PC2     INCT 14                 R14=CORRECT PC ADDR. FOR RTWP
005C C1D2          MOV  *2,7               SAVE CURRENT PULSE COUNT
005E 8147          C    7,5                IS CURRENT COUNT A MAXIMUM?
0060 1201          JLE  S1                 NO,JMP TO TIME OVERFLOW TEST
0062 C147          MOV  7,5                YES, SAVE MAX. COUNT IN R5
0064 8047  S1      C    7,1                IS COUNT GT ALLOWED MAX.?
0066 1A01          JL   DELAY              NO, JMP TO WAIT LOOP
0068 0586          INC  6                  YES, INC R6 = NO. OF OVERFLOWS
006A 8052  DELAY   C    *2,1               HAS TIME EXPIRED ?
006C 1AFE          JL   DELAY              NO, WAIT FOR INTERRUPTS
006E 04D2          CLR  *2                 YES, RESET COUNTER
0070 0380          RTWP                    RETURN
             *
             *
             *  SUBROUTINE TIMEOF
             *  STOP THE CLOCK AND RETURN ARGUMENTS TO CALLING PROGRAM
             *
0072 0004' TIMEOF DATA WSP1                SUB. TIMEOF WORKSPACE ADDR.
0074 0076'         DATA PC3                SUB. TIMEOF PC ADDR.
0076 0300  PC3     LIMI 0
0078 0000
007A 03C0          CKOF                    STOP THE CLOCK
007C 0585          INC  5                  R5=R5+1=MAX COUNT NEEDED BY
             *                             RUNTIME ROUTINES
007E 05CE          INCT 14                 *14=ADDR OF 1ST ARGUMENT
0080 C23E          MOV  *14+,8             R8=ADDR OF 1ST ARGUMENT
0082 C27E          MOV  *14+,9             R9=ADDR OF 2ND ARGUMENT
             *                             R14=CORRECT PC ADDR. FOR RTWP
0084 C605          MOV  5,*8               1ST ARGUMENT = MAX COUNT
0086 C646          MOV  6,*9               2ND ARGUMENT = NO. OF TIME
             *                             OVERFLOWS IN RUNTIME LOOP
0088 0300          LIMI 5
008A 0005
008C 0380          RTWP                    RETURN
             *
             *
             *  ROUTINE CLK5
             *  CLOCK INTERRUPT PROCESSING ROUTINE
             *
```

```
0099 008E       WSP    BSS  32              WORKSPACE FOR CLK5
0100 00AE 0580  CLK5   INC  0               COUNT CLOCK PULSES
0101 00B0 03C0         CKOF                 CLEAR THE INTERRUPT
0102 00B2 03A0         CKON                 RESTART THE CLOCK
0103 00B4 0380         RTWP                 RETURN
0104                   END
```

```
 ' CLK5     00AE    ' DELAY   006A    ' PC1     0024    ' PC2     005A
 ' PC3      0076    ' S1      0064    D TIMEOF  0072    D TIMEON  0000
 D WAIT     0056    ' WSP     008E    ' WSP1    0004

 0000 ERRORS
```

```
                        IDT   'INPCAS'           MAY 20,1979
                 *
                 *  THIS MODULE CONTAINS 8 FORTRAN CALLABLE ROUTINES
                 *  THAT READ ASCII DATA FROM TI 733ASR CASSETTES 1 AND
                 *  2(LUNO 7 AND 8). FOUR ENTRY POINTS ARE SUPPLIED
                 *  FOR EACH CASSETTE WHICH ALLOWS READING (1)INTEGER,
                 *  (2)EXTENDED INTEGER, (3)REAL, AND (4)LOGICAL DATA
                 *  TYPES. THE ASCII CHARACTERS ARE CONVERTED TO BINARY
                 *  WORDS(4 CHAR PER WORD) AND STORED IN CONSECUTIVE
                 *  MEMORY LOCATIONS BEGINNING AT THE ADDRESS IN "ARG1".
                 *  -- ONE ARGUMENT USED --
                 *     CASSETTE 1 ROUTINES READ MULTIPLE RECORDS. THE FIRST
                 *  RECORD ON TAPE SHOULD CONTAIN THE RECORD COUNT (1ST 4
                 *  CHARACTERS). REMAINING CHARS. IN THE 1ST RECORD ARE
                 *  IGNORED. THE REMAINING RECORDS SHOULD CONTAIN INPUT DATA
                 *  PACKED UP TO 80 CHARS. PER RECORD (= 20 COMPUTER WORDS).
                 *     CASSETTE 2 ROUTINES READ ONLY 1 RECORD PER CALL. THE
                 *  RECORD SHOULD CONTAIN INPUT DATA PACKED UP TO 80 CHARS.
                 *  PER RECORD.
                 *
                 *  EXAMPLE:  CALL INCS2I(J)     WILL READ A RECORD FROM
                 *            CASSETTE 2 AND STORE ALL WORDS IN THAT RECORD
                 *            STARTING AT LOCATION "J"
                 *
                        DEF  INCS1I           CASSETTE 1 INTEGER INPUT
                        DEF  INCS1E           CASSETTE 1 EXTENDED INT. INPUT
                        DEF  INCS1R           CASSETTE 1 REAL INPUT
                        DEF  INCS1L           CASSETTE 1 LOGICAL INPUT
                 *
                        DEF  INCS2I           CASSETTE 2 INTEGER INPUT
                        DEF  INCS2E           CASSETTE 2 EXTENDED INT. INPUT
                        DEF  INCS2R           CASSETTE 2 REAL INPUT
                        DEF  INCS2L           CASSETTE 2 LOGICAL INPUT
                 *
0000 008E'  INCS1I DATA WSP                   WSP = ADDR. OF WORKSPACE
0002 0020'         DATA START1                START1 = ADDR. OF PC FOR CS1
0004 008E'  INCS1E DATA WSP
0006 0020'         DATA START1
0008 008E'  INCS1R DATA WSP
000A 0020'         DATA START1
000C 008E'  INCS1L DATA WSP
000E 0020'         DATA START1
                 *
0010 008E'  INCS2I DATA WSP
0012 0064'         DATA START2                START2= ADDR.OF PC FOR CS2
0014 008E'  INCS2E DATA WSP
0016 0064'         DATA START2
0018 008E'  INCS2R DATA WSP
001A 0064'         DATA START2
001C 008E'  INCS2L DATA WSP
001E 0064'         DATA START2
                 *
                 *
                 *  READ FROM CASSETTE 1 (LUNO 7) - MULTIPLE RECORDS
```

```
                  *
     0020 2FE0  START1 XOP   @CS1IC,15         OPEN LUNO 7 (CASSETTE 1)
     0022 00FE'
     0024 2FE0         XOP   @INPT1,15         READ 1ST TAPE RECORD = NO. OF
     0026 010A'
                  *                            RECORDS THAT FOLLOW
     0028 C820         MOV   @BUF,@W1          MOVE 1ST 4 CHARS. INTO
     002A 00AE'
     002C 0130'
     002E C820         MOV   @BUF+2,@W2        CONVERSION AREA
     0030 00B0'
     0032 0132'
     0034 2FE0         XOP   @ASKBIN,15        CONVERT ASCII TO BINARY
     0036 012E'
                  *                            R0 = NO. OF RECS. ON TAPE
     0038 C040         MOV   0,1               R1= NO. OF RECS. ON TAPE
     003A 05CE         INCT  14                *14 = ADDR. OF 1ST STORAGE LOC
     003C C0BE         MOV   *14+,2            R2 = ADDR. OF 1ST STORAGE LOC.
                  *                            R14 = CORRECT PC ADDR. FOR RTW
     003E 2FE0  S1     XOP   @INPT1,15         READ A RECORD FROM LUNO 7
     0040 010A'
     0042 C0E0         MOV   @NCHAR1,3         R3 = CHARACTER COUNT(THIS REC)
     0044 0114'
     0046 0923         SRL   3,2               R3 = WORD COUNT(THIS RECORD)
     0048 0204         LI    4,BUF             R4 = ADDR FOR ASCII STORAGE
     004A 00AE'
     004C C834  S2     MOV   *4+,@W1           -- MOVE 4 ASCII CHARACTERS
     004E 0130'
     0050 C834         MOV   *4+,@W2           INTO CONVERSION AREA --
     0052 0132'
     0054 2FE0         XOP   @ASKBIN,15        CONVERT 4 ASCII CHARS. INTO
     0056 012E'
                  *                            ONE BINARY WORD (R0)
     0058 CC80         MOV   0,*2+             TRANSFER BINARY WORD TO MEMORY
     005A 0603         DEC   3                 ARE ALL WORDS IN CURRENT
                  *                            RECORD CONVERTED ?
     005C 1BF7         JH    S2                NO, TRANSFER AND STORE AGAIN
     005E 0601         DEC   1                 HAVE ALL RECORDS BEEN READ ?
     0060 1BEE         JH    S1                NO, READ ANOTHER RECORD
     0062 0380         RTWP                    YES, RETURN
                  *
                  *
                  *  READ FROM CASSETTE 2 (LUNO 8) SINGLE RECORD
                  *
     0064 2FE0  START2 XOP   @CS2IC,15         OPEN LUNO 8(CASSETTE 2)
     0066 0116'
     0068 05CE         INCT  14                *14=ADDR. OF 1ST STORAGE LOC.
     006A C0BE         MOV   *14+,2            R2=ADDR OF 1ST STORAGE LOC.
                  *                            R14=CORRECT PC ADDR.FOR RTWP
     006C 2FE0         XOP   @INPT2,15         READ THE CASSETTE RECORD
     006E 0122'
     0070 C0E0         MOV   @NCHAR2,3         R3 = CHARACTER COUNT
     0072 012C'
     0074 0923         SRL   3,2               R3 = WORD COUNT
```

```
0076 0204          LI   4,BUF               R4 = ADDR FOR ASCII STORAGE
0078 00AE'
007A C834   S3     MOV  *4+,@W1             -- MOVE 4 ASCII CHARACTERS
007C 0130'
007E C834          MOV  *4+,@W2             INTO CONVERSION AREA
0080 0132'
0082 2FE0          XOP  @ASKBIN,15          CONVERT 4 ASCII CHARS. INTO
0084 012E'
            *                               ONE BINARY WORD (R0)
0086 CC80          MOV  0,*2+               TRANSFER BINARY WORD TO MEMORY
0088 0603          DEC  3                   ARE ALL WORDS IN RECORD
            *                               CONVERTED AND STORED ?
008A 1BF7          JH   S3                  NO, TRANSFER AND STORE AGAIN
008C 0380          RTWP                     YES, RETURN
            *
            *
            *
008E        WSP    BSS  32                  WORKSPACE AREA
00AE        BUF    BSS  80                  STORAGE FOR ASCII CHARACTERS
            *
            ***** XOP 15 DATA BLOCKS *****
            *
00FE 0000   CS1IC  DATA 0,7                 OPEN LUNO 7(CASSETTE 1)
0100 0007
0102               BSS  8
            *
010A 0000   INPT1  DATA 0,>0907,0,BUF,80  READ ASCII REC. FROM LUNO 7
010C 0907
010E 0000
0110 00AE'
0112 0050
0114 0000   NCHAR1 DATA 0
            *
0116 0000   CS2IC  DATA 0,8                 OPEN LUNO 8(CASSETTE 2)
0118 0008
011A               BSS  8
            *
0122 0000   INPT2  DATA 0,>0908,0,BUF,80  READ ASCII REC. FROM LUNO 8
0124 0908
0126 0000
0128 00AE'
012A 0050
012C 0000   NCHAR2 DATA 0
            *
012E 0D00   ASKBIN DATA >0D00               CONVERT ASCII TO BINARY
0130 0000   W1     DATA 0
0132 0000   W2     DATA 0
            *
                   END
```

'	ASKBIN	012E	'	BUF	00AE	'	CS1IC	00FE	'	CS2IC	0116
D	INCS1E	0004	D	INCS1I	0000	D	INCS1L	000C	D	INCS1R	0008
D	INCS2E	0014	D	INCS2I	0010	D	INCS2L	001C	D	INCS2R	0018
'	INPT1	010A	'	INPT2	0122	'	NCHAR1	0114	'	NCHAR2	012C
'	S1	003E	'	S2	004C	'	S3	007A	'	START1	0020
'	START2	0064	'	W1	0130	'	W2	0132	'	WSP	008E

0000 ERRORS

```
                         IDT  'IOASR'           JUNE 15,1979
                    *
                    *  THIS MODULE CONTAINS 8 FORTRAN CALLABLE ROUTINES THAT
                    *  PERFORM INPUT AND OUTPUT TO THE TI 733ASR TELETYPE
                    *  (LUNO 6). FOUR ENTRY POINTS FOR INPUT AND FOUR FOR
                    *  OUTPUT ALLOW READING AND WRITING (1)INTEGER, (2)EXTENDED
                    *  INTEGER, (3)REAL, AND (4)LOGICAL DATA TYPES. THREE
                    *  ARGUMENTS ARE USED:
                    *     (1)ARG1 = STARTING ADDR OF DATA TO BE INPUT OR OUTPUT.
                    *     (2)ARG2 = ADDRESS OF THE NUMBER OF CONSECUTIVE MEMORY
                    *               WORDS TO BE INPUT OR OUTPUT.
                    *     (3)ARG3 = STARTING ADDRESS OF THE TEXT. INPUT
                    *               ROUTINES REQUIRE CONSECUTIVE 6 CHARACTER
                    *               TEXTS FOR EACH WORD INPUT. OUTPUT ROUTINES
                    *               REQUIRE ONLY ONE 6 CHARACTER TEXT FOR EACH
                    *               SUBROUTINE CALL.
                    *  INTEGER*2 INPUTS AND OUTPUTS ARE DECIMAL NUMBERS.
                    *  ALL OTHER INPUTS AND OUTPUTS ARE HEXADECIMAL NUMBERS
                    *
                    *     EXAMPLES:  CALL PRASRI(J,8,JTEXT)   WILL PRINT 8 INTEG
                    *                NUMBERS STARTING AT LOCATION "J".
                    *                   USE:  DIMENSION JTEXT(3)    AND
                    *                         DATA JTEXT/6H      J/
                    *
                    *                CALL INASRR(A,6,ITEXT)  WILL READ 3 REAL
                    *                NUMBERS (6 MEMORY WORDS) AND STORE THE
                    *                BINARY STARTING AT LOCATION "A".
                    *                   USE:  DIMENSION ITEXT(18)    AND
                    *                         DATA ITEXT/6H     A,30H     .../
                    *
                    *
                         DEF  INASRI            INPUT FROM TTY - INTEGER
                         DEF  INASRE            INPUT FROM TTY - EXTENDED INT.
                         DEF  INASRR            INPUT FROM TTY - REAL
                         DEF  INASRL            INPUT FROM TTY - LOGICAL
                    *
                         DEF  PRASRI            PRINT TO TTY - INTEGER
                         DEF  PRASRE            PRINT TO TTY - EXTENDED INT.
                         DEF  PRASRR            PRINT TO TTY - REAL
                         DEF  PRASRL            PRINT TO TTY - LOGICAL
                    *
                    *
0000 0020'  INASRI DATA WSP                     WSP = ADDR. OF WORKSPACE
0002 0052'         DATA START3                  START3 = PC FOR DECIMAL INPUTS
0004 0020'  INASRE DATA WSP
0006 0040'         DATA START1                  START1 = PC FOR HEX INPUTS
0008 0020'  INASRR DATA WSP
000A 0040'         DATA START1
000C 0020'  INASRL DATA WSP
000E 0040'         DATA START1
                    *
0010 0020'  PRASRI DATA WSP
0012 00AC'         DATA START4                  START4 = PC FOR DECIMAL OUTPUT
0014 0020'  PRASRE DATA WSP
```

```
0016 009A'        DATA START2             START2 = PC FOR HEX OUTPUT
0018 0020' PRASRR DATA WSP
001A 009A'        DATA START2
001C 0020' PRASRL DATA WSP
001E 009A'        DATA START2
           *
0020       WSP    BSS  32                 WORKSPACE
           *
           *
           *  READ FROM TTY (LUNO 6) AND STORE DATA IN MEMORY
           *
0040 0206  START1 LI   6,>0D00
0042 0D00
0044 C806         MOV  6,@CTB             CONVERT HEX TO BINARY
0046 0178'
0048 0206         LI   6,4
004A 0004
004C C806         MOV  6,@RVAL+8
004E 0174'
0050 1008         JMP  S5
0052 0206  START3 LI   6,>0B00
0054 0B00
0056 C806         MOV  6,@CTB             CONVERT DEC. TO BINARY
0058 0178'
005A 0206         LI   6,6
005C 0006
005E C806         MOV  6,@RVAL+8
0060 0174'
0062 05CE  S5     INCT 14
0064 C07E         MOV  *14+,1             R1 = ADDR. OF WORD
0066 C0BE         MOV  *14+,2             R2 = ADDR. OF WORD COUNT
0068 C17E         MOV  *14+,5             R5 = ADDR. OF TEXT
           *                              R14= CORRECT PC ADDR FOR RTWP
006A 2FE0         XOP  @LFCR,15           LINE FEED + CR
006C 0128'
006E C0D2         MOV  *2,3               R3 = WORD COUNT
0070 1213         JLE  RETURN             RETURN IF WORD COUNT NOT +
0072 C835  S1     MOV  *5+,@CHAR          CHAR CONTAINS TEXT CHARACTERS
0074 0190'
0076 C835         MOV  *5+,@CHAR+2
0078 0192'
007A C835         MOV  *5+,@CHAR+4
007C 0194'
007E 2FE0         XOP  @PTEXT,15          PRINT TEXT
0080 0196'
0082 2FE0         XOP  @EQUALS,15         PRINT " = "
0084 0184'
0086 2FE0         XOP  @RVAL,15           READ HEX ASCII VALUE
0088 016C'
008A 2FE0         XOP  @CTB,15            ACSII TO BINARY CONVERSION
008C 0178'
008E CC40         MOV  0,*1+              STORE BINARY WORD
0090 2FE0         XOP  @LFCR,15           LINE FEED + CR
0092 0128'
```

```
0092 0094 0603          DEC  3                  HAVE ALL WORDS BEEN TAKEN ?
0093 0096 16ED          JNE  S1                 NO, READ ANOTHER WORD
0094 0098 0380   RETURN RTWP                    YES, RETURN
0095             *
0096             *
0097             *  PRINT TO TTY (LUNO 6)
0098             *
0099 009A 0206   START2 LI   6,>0C00
     009C 0C00
0100 009E C806          MOV  6,@CTA             CONVERT BINARY TO HEX ASCII
     00A0 0158'
0101 00A2 0206          LI   6,4
     00A4 0004
0102 00A6 C806          MOV  6,@PVAL+10
     00A8 016A'
0103 00AA 1008          JMP  S6
0104 00AC 0206   START4 LI   6,>0A00
     00AE 0A00
0105 00B0 C806          MOV  6,@CTA             CONVERT BINARY TO DEC. ASCII
     00B2 0158'
0106 00B4 0206          LI   6,6
     00B6 0006
0107 00B8 C806          MOV  6,@PVAL+10
     00BA 016A'
0108 00BC 05CE   S6     INCT 14
0109 00BE C07E          MOV  *14+,1             R1 = ADDR. OF WORD
0110 00C0 C0BE          MOV  *14+,2             R2 = ADDR. OF WORD COUNT
0111 00C2 C17E          MOV  *14+,5             R5 = ADDR. OF TEXT
0112             *                              R14= CORRECT PC ADDR FOR RTWP
0113 00C4 C0D2          MOV  *2,3               R3 = WORD COUNT
0114 00C6 121C          JLE  S4                 LINE FEED + CR IF WORD CNT.=0
0115 00C8 C835          MOV  *5+,@CHAR          CHAR CONTAINS TEXT CHARACTERS
     00CA 0190'
0116 00CC C835          MOV  *5+,@CHAR+2
     00CE 0192'
0117 00D0 C835          MOV  *5+,@CHAR+4
     00D2 0194'
0118 00D4 2FE0          XOP  @PTEXT,15          PRINT TEXT
     00D6 0196'
0119 00D8 2FE0          XOP  @EQUALS,15         PRINT " = "
     00DA 0184'
0120 00DC 0204   S2     LI   4,8                R4 = WORDS PER LINE COUNTER
     00DE 0008
0121 00E0 C031   S3     MOV  *1+,0              R0 = CONVERSION AREA
0122 00E2 2FE0          XOP  @CTA,15            BINARY TO ASCII
     00E4 0158'
0123 00E6 2FE0          XOP  @PVAL,15           PRINT VALUE
     00E8 0160'
0124 00EA 2FE0          XOP  @SPACE2,15         PRINT 2 SPACES
     00EC 0136'
0125 00EE 0603          DEC  3                  HAVE ALL VALUES BEEN PRINTED ?
0126 00F0 1207          JLE  S4                 YES, RETURN
0127 00F2 0604          DEC  4                  DOES CURRENT LINE CONTAIN
0128             *                              8 VALUES ?
```

```
0129 00F4 15F5          JGT  S3                 NO PRINT NEXT VALUE ON
0130                *                           SAME LINE
0131 00F6 2FE0          XOP  @LFCR,15           YES, START A NEW LINE,
     00F8 0128'
0132 00FA 2FE0          XOP  @SPACE9,15         INDENT, AND
     00FC 014C'
0133 00FE 10EE          JMP  S2                 RESUME PRINTING
0134 0100 2FE0     S4   XOP  @LFCR,15           LINE FEED + CR
     0102 0128'
0135 0104 0380          RTWP                    RETURN
0136                *
0137                *
0138 0106               BSS  32                 WORKSPACE
0139                *
0140                ******  XOP 15 DATA  *****
0141                *
0142 0126 0D0A     D1     DATA >0D0A
0143 0128 0000     LFCR   DATA 0,>0B06,0,D1,0,2  LINE FEED + CR
     012A 0B06
     012C 0000
     012E 0126'
     0130 0000
     0132 0002
0144                *
0145 0134 2020     D2     DATA >2020
0146 0136 0000     SPACE2 DATA 0,>0B06,0,D2,0,2  PRINT 2 SPACES
     0138 0B06
     013A 0000
     013C 0134'
     013E 0000
     0140 0002
0147                *
0148 0142 2020     D3     DATA >2020,>2020,>2020
     0144 2020
     0146 2020
0149 0148 2020            DATA >2020,>2020
     014A 2020
0150 014C 0000     SPACE9 DATA 0,>0B06,0,D3,0,9  PRINT 9 SPACES
     014E 0B06
     0150 0000
     0152 0142'
     0154 0000
     0156 0009
0151                *
0152 0158          CTA    BSS  2                CONVERT ASCII TO BINARY
0153 015A          AOUT   BSS  6
0154                *
0155 0160 0000     PVAL   DATA 0,>0B06,0,AOUT    PRINT ASCII VALUE
     0162 0B06
     0164 0000
     0166 015A'
0156 0168                 BSS  4
0157                *
0158 016C 0000     RVAL   DATA 0,>0906,0,AIN     READ ASCII VALUE
```

```
      016E 0906
      0170 0000
      0172 017A'
 0159 0174               BSS  4
 0160               *
 0161 0178          CTB    BSS  2                  CONVERT ASCII TO BINARY
 0162 017A          AIN    BSS  6
 0163               *
 0164 0180 203D     D4     DATA >203D,>2020
      0182 2020
 0165 0184 0000     EQUALS DATA 0,>0B06,0,D4,0,3  PRINT " = "
      0186 0B06
      0188 0000
      018A 0180'
      018C 0000
      018E 0003
 0166               *
 0167 0190          CHAR   BSS  6
 0168 0196 0000     PTEXT  DATA 0,>0B06,0,CHAR,0,6  PRINT TEXT
      0198 0B06
      019A 0000
      019C 0190'
      019E 0000
      01A0 0006
 0169               *
 0170               *
 0171                      END
```

```
 ' AIN     017A   ' AOUT    015A   ' CHAR    0190   ' CTA     0158
 ' CTB     0178   ' D1      0126   ' D2      0134   ' D3      0142
 ' D4      0180   ' EQUALS  0184   D INASRE  0004   D INASRI  0000
 D INASRL  000C   D INASRR  0008   ' LFCR    0128   D PRASRE  0014
 D PRASRI  0010   D PRASRL  001C   D PRASRR  0018   ' PTEXT   0196
 ' PVAL    0160   ' RETURN  0098   ' RVAL    016C   ' S1      0072
 ' S2      00DC   ' S3      00E0   ' S4      0100   ' S5      0062
 ' S6      00BC   ' SPACE2  0136   ' SPACE9  014C   ' START1  0040
 ' START2  009A   ' START3  0052   ' START4  00AC   ' WSP     0020

0000 ERRORS
```

```
                      IDT   'I09300'          JUNE 5,1979
               *
               *  THIS MODULE CONTAINS FORTRAN CALLABLE ROUTINES WHICH
               *  PERFORM INPUT AND OUTPUT DATA TRANSFERS BETWEEN THE
               *  TI-990 AND THE SDS-9300 COMPUTERS. 2 ARGUMENTS ARE
               *  USED IN THE CALLING SEQUENCE:
               *     (1)ARG1 = STARTING ADDRESS OF THE MEMORY BLOCK
               *  FROM WHICH TRANSMITTED DATA IS SENT OR WHERE
               *  RECEIVED DATA IS STORED.
               *     (2)ARG2 = ADDRESS OF THE NUMBER OF CONSECUTIVE
               *  MEMORY WORDS TO BE TRANSFERRED (IE. ADDR. OF THE
               *  WORD COUNT).
               *
               *     SUBROUTINE R9300I   RECEIVES 14 BIT INTEGER (2'S
               *  COMPLEMENT) DATA FROM THE INTERFACE AND STORES THE
               *  DATA IN TI-990 16 BIT MEMORY WORDS. THE INPUT DATA
               *  IS PLACED IN THE MOST SIGNIFICANT 14 BITS OF THE
               *  MEMORY WORD (WITH 2 LSB ZERO FILLED).
               *
               *     SUBROUTINE W9300I   TRANSMITS 14 BIT INTEGER (2'S
               *  COMPLEMENT) DATA TO THE INTERFACE. THE 14 MOST
               *  SIGNIFICANT BITS OF THE TI-990 WORD ARE TRANSMITTED.
               *
               *     SUBROUTINE W9300R  TRANSMITS REAL DATA TO THE
               *  INTERFACE. TI-990 FLOATING POINT WORDS ARE SENT IN
               *  TWO 14 BIT TRANSFERS WITH A SUBSEQUENT LOSS OF 4 BITS
               *  OF MAGNITUDE PRECISION. (1)1ST 14 BIT TRANSFER: BIT 0
               *  (MSB) IS THE MAGNITUDE SIGN BIT, BITS 1-7 ARE EXPONENT
               *  BITS IN EXCESS 64 NOTATION (BIASED BY >40), BITS 8-13
               *  ARE THE 6 MOST SIGNIFICANT DATA MAGNITUDE BITS. (2)2ND
               *  14 BIT TRANSFER: BITS 0-13 ARE THE LOWER ORDER MAGNI-
               *  TUDE BITS. THE 9300 MUST USE THIS INFORMATION TO
               *  CONSTRUCT ITS OWN FLOATING POINT WORD.
               *
               *  EXAMPLES:  CALL R9300I(ISTART,10)  TAKES 10 INTEGER
               *           WORDS FROM THE INTERFACE AND STORES THEM IN
               *           CONSECUTIVE MEMORY STARTING AT "ISTART".
               *
               *             CALL W9300R(ASTART,10)  PUTS 5 CONSECU-
               *           TIVE REAL WORDS (10 CONSEC. MEMORY WORDS) ON
               *           THE INTERFACE BEGINNING AT "ASTART".
               *
               *
                      DEF   R9300I            READ INTEGER DATA
                      DEF   W9300I            WRITE INTEGER DATA
                      DEF   W9300R            WRITE REAL DATA
               *
               *
               *
0000 000C' R9300I DATA WSP                    WORKSPACE
0002 002C'        DATA READI
0004 000C' W9300I DATA WSP
0006 004A'        DATA WRITEI
0008 000C' W9300R DATA WSP
```

```
     000A 006A'         DATA WRITER
                *
     000C       WSP     BSS  20             WORKSPACE R0-R9
     0020 3FFF          DATA >3FFF          R10 = INITIALIZE CRU TO 3FFF
     0022 FFFC          DATA >FFFC          R11 = MASK 14 MSB
     0024 0020          DATA >20            R12 = CRU BASE ADDR. (>20)
     0026               BSS  6              WORKSPACE R13-R15
                *
                *  SUBROUTINE R9300I - INTEGER READ
     002C 05CE  READI   INCT 14             *14= ADDR. OF WORD
     002E C07E          MOV  *14+,1         R1 = ADDR. OF WORD
     0030 C0BE          MOV  *14+,2         R2 = ADDR. OF WORD COUNT
                *                           R14= CORRECT PC ADDR. FOR RTWP
     0032 C0D2          MOV  *2,3           R3 = WORD COUNT
     0034 1E0F          SBZ  15             ENABLE 9300
     0036 1F0F  S1      TB   15             IS 9300  OUTPUT READY ?
     0038 13FE          JEQ  S1             NO, WAIT
     003A 1E0E          SBZ  14             YES, ARM THE INTERFACE
     003C 3784          STCR 4,14           READ THE INTERFACE
     003E 1D0E          SBO  14             SEND WORD RECEIVED SIGNAL
     0040 0A24          SLA  4,2            LEFT JUSTIFY THE WORD
     0042 CC44          MOV  4,*1+          STORE THE WORD
     0044 0603          DEC  3              ALL WORDS RECEIVED ?
     0046 15F7          JGT  S1             NO, RECEIVE ANOTHER WORD
     0048 0380          RTWP                YES, RETURN
                *
                *
                *  SUBROUTINE W9300I - INTEGER WRITE
     004A 05CE  WRITEI  INCT 14             *14= ADDR. OF WORD
     004C C07E          MOV  *14+,1         R1 = ADDR. OF WORD
     004E C0BE          MOV  *14+,2         R2 = ADDR. OF WORD COUNT
                *                           R14= CORRECT PC ADDR. FOR RTWP
     0050 C0D2          MOV  *2,3           R3 = WORD COUNT
     0052 338A          LDCR 10,14          INITILIZE CRU
     0054 1E0E          SBZ  14             ENABLE THE 9300
     0056 C131  S5      MOV  *1+,4          R4 = 990 WORD
     0058 0924          SRL  4,2            R4 = 14 BITS FOR THE 9300
     005A 1F0E  S6      TB   14             IS 9300 INPUT READY ?
     005C 13FE          JEQ  S6             NO, WAIT
     005E 1E0F          SBZ  15             YES, ARM THE INTERFACE
     0060 3384          LDCR 4,14           SEND THE WORD
     0062 1D0F          SBO  15             SEND WORD SENT SIGNAL
     0064 0603          DEC  3              ALL WORDS SENT ?
     0066 15F7          JGT  S5             NO, SEND ANOTHER WORD
     0068 0380          RTWP                YES, RETURN
                *
                *
                *  SUBROUTINE W9300R - REAL WRITE
     006A 05CE  WRITER  INCT 14             *14= ADDR. OF WORD
     006C C07E          MOV  *14+,1         R1 = ADDR. OF WORD
     006E C0BE          MOV  *14+,2         R2 = ADDR. OF WORD COUNT
                *                           R14= CORRECT PC ADDR. FOR RTWP
     0070 C0D2          MOV  *2,3           R3 = WORD COUNT
     0072 338A          LDCR 10,14          INITIALIZE CRU
```

```
I09300        TXMIRA        2.3.0  78.244 09:43:46    07/10/79     PAGE 0003

0074 1E0E              SBZ  14                      ENABLE THE 9300
0076 C131      S2      MOV  *1+,4                   R4 = 1ST WORD OF 990 F.P.
0078 C171              MOV  *1+,5                   R5 = 2ND WORD OF 990 F.P.
007A C184              MOV  4,6
007C 418B              SZC  11,6                    R6 = SAVE 2 LSB OF 1ST WORD
007E E146              SOC  6,5
0080 0924              SRL  4,2                     R4 = 14 BITS FOR 9300   WORD 1
0082 0B45              SRC  5,4                     R5 = 14 BITS FOR 9300   WORD 2
0084 1F0E      S3      TB   14                      IS 9300 INPUT READY ?
0086 13FE              JEQ  S3                      NO, WAIT
0088 1E0F              SBZ  15                      YES, ARM THE INTERFACE
008A 3384              LDCR 4,14                    SEND WORD 1
008C 1D0F              SBO  15                      SEND WORD SENT SIGNAL
008E 1F0E      S4      TB   14                      IS 9300 INPUT READY ?
0090 13FE              JEQ  S4                      NO, WAIT
0092 1E0F              SBZ  15                      YES, ARM THE INTERFACE
0094 3385              LDCR 5,14                    SEND WORD 2
0096 1D0F              SBO  15                      SEND WORD SENT SIGNAL
0098 0643              DECT 3                       ALL F.P. WORDS SENT ?
009A 15ED              JGT  S2                      NO, SEND ANOTHER F.P. WORD
009C 0380              RTWP                         YES, RETURN
               *
                       END
```

```
 D R9300I  0000     ' READI   002C     ' S1      0036     ' S2      0076
 ' S3      0084     ' S4      008E     ' S5      0056     ' S6      005A
 D W9300I  0004     D W9300R  0008     ' WRITEI  004A     ' WRITER  006A
 ' WSP     000C
```

0000 ERRORS

```
                    IDT  'OUTCAS'          MAY 29,1979
               *
               *  THIS MODULE CONTAINS 4 FORTRAN CALLABLE ROUTINES THAT
               *  WRITE DATA TO THE TI 733ASR CASSETTE 1 (LUNO 7). A
               *  FIFTH ROUTINE WRITES AN END OF FILE MARK ON CASSETTE
               *  1 (LUNO 7).
               *  4 DATA WRITE ENTRY POINTS ARE SUPPLIED WHICH ALLOW
               *  WRITING (1)INTEGER, (2)EXTENDED INTEGER, (3)REAL,
               *  AND (4)LOGICAL DATA TYPES. BINARY DATA IN MEMORY IS
               *  FIRST CONVERTED TO HEXADECIMAL ASCII CHARACTERS (4
               *  CHARS. PER WORD) AND THEN OUTPUT TO THE CASSETTE
               *  UNIT. EACH SUBROUTINE CALL WILL GENERATE ONE RECORD.
               *  THE USER SHOULD BE AWARE THAT A MAXIMUM OF 20 MEMORY
               *  WORDS (80 ASCII CHARS.) MAY BE WRITTEN IN EACH
               *  CASSETTE RECORD. TWO ARGUMENTS ARE USED WITH THE 4
               *  DATA WRITE ROUTINES :
               *     (1)ARG1 = STARTING ADDRESS OF DATA TO BE OUTPUT.
               *     (2)ARG2 = ADDRESS OF THE NUMBER OF CONSECUTIVE
               *               MEMORY WORDS TO BE OUTPUT (ADDR. OF
               *               THE 'WORD COUNT').
               *  NO ARGUMENTS ARE USED WITH THE EOF WRITE ROUTINE.
               *
               *  EXAMPLES:  CALL OUCS1I(I,3)      WILL WRITE A RECORD
               *             ON CASSETTE 1 CONTAINING THE HEX. ASCII
               *             CHARACTER REPRESENTATION OF 3 CONSECUTIVE
               *             MEMORY WORDS BEGINNING AT LOCATION "I".
               *
               *             CALL OUCS1R(A,8)      WILL WRITE A RECORD
               *             ON CASSETTE 1 CONTAINING THE HEX ASCII
               *              CHARACTER REPRESENTATION OF 4 CONSECUTIVE
               *             REAL WORDS (8 CONSEC. MEMORY WORDS)
               *             BEGINNING AT LOCATION "A".
               *
               *             CALL EOFCS1      WILL WRITE AN END OF FILE
               *             MARK ON CASSETTE 1.
               *
                      DEF  OUCS1I              CS1 INTEGER OUTPUT
                      DEF  OUCS1E              CS1 EXTENDED INTEGER OUTPUT
                      DEF  OUCS1R              CS1 REAL OUTPUT
                      DEF  OUCS1L              CS1 LOGICAL OUTPUT
                      DEF  EOFCS1              CS1 WRITE EOF
               *
0000 0014'     OUCS1I DATA WSP
0002 0084'            DATA START1
0004 0014'     OUCS1E DATA WSP
0006 0084'            DATA START1
0008 0014'     OUCS1R DATA WSP
000A 0084'            DATA START1
000C 0014'     OUCS1L DATA WSP
000E 0084'            DATA START1
0010 0014'     EOFCS1 DATA WSP
0012 00BE'            DATA END1
               *
0014           WSP    BSS  32
```

```
 0055 0034          BUF    BSS  80
 0056               *
 0057               *  WRITE TO CASSETTE 1 (LUNO 7) - ONE RECORD
 0058               *
 0059 0084 05CE     START1 INCT 14                    *14=ADDR. OF 1ST WORD
 0060 0086 C07E            MOV  *14+,1                R1 = ADDR. OF 1ST WORD
 0061 0088 C0BE            MOV  *14+,2                R2 = ADDR. OF WORD COUNT
 0062               *                                 R14= CORRECT PC ADDR FOR RTWP
 0063 008A C0D2            MOV  *2,3                  R3 = WORD COUNT
 0064 008C 0283            CI   3,20                  TOO MANY WORDS IN RECORD ?
      008E 0014
 0065 0090 1202            JLE  S1                    NO, CONTINUE
 0066 0092 0203            LI   3,20                  YES,SET WORD CNT TO MAX. (20)
      0094 0014
 0067 0096 2FE0     S1     XOP  @CS1IC,15             OPEN LUNO 7 (CS1)
      0098 00CA'
 0068 009A C103            MOV  3,4                   R4 = WORD COUNT
 0069 009C 0A24            SLA  4,2                   R4 = CHARACTER COUNT
 0070 009E 0205            LI   5,BUF                 R5 = ADDR OF CHARACTER BUFFER
      00A0 0034'
 0071 00A2 C031     S2     MOV  *1+,0                 MOVE BINARY WORD TO R0
 0072 00A4 2FE0            XOP  @BINASK,15            CONVERT BINARY(R0) TO ASCII
      00A6 00D6'
 0073 00A8 CD60            MOV  @W1,*5+               STORE ASCII(W1,W2) IN
      00AA 00D8'
 0074 00AC CD60            MOV  @W2,*5+               CHARACTER BUFFER
      00AE 00DA'
 0075 00B0 0603            DEC  3                     ALL WORDS CONVERTED ?
 0076 00B2 1BF7            JH   S2                    NO, CONVERT ANOTHER WORD
 0077 00B4 C804            MOV  4,@NCHAR1             YES, LOAD CHARACTER COUNT
      00B6 00E6'
 0078 00B8 2FE0            XOP  @OUTP1,15             WRITE THE RECORD ON LUNO 7
      00BA 00DC'
 0079 00BC 0380            RTWP                       RETURN
 0080               *
 0081               *  WRITE END OF FILE ON LUNO 7 (CASSETTE 1)
 0082               *
 0083 00BE 2FE0     END1   XOP  @CS1IC,15
      00C0 00CA'
 0084 00C2 2FE0            XOP  @WEOF1,15             WRITE EOF ON CS1
      00C4 00E8'
 0085 00C6 05CE            INCT 14                    R14= CORRECT PC AADR FOR RTWP
 0086 00C8 0380            RTWP                       RETURN
 0087               *
 0088               *****  XOP 15 DATA  *****
 0089               *
 0090 00CA 0000     CS1IC  DATA 0,7                   OPEN LUNO 7 (CASSETTE 1)
      00CC 0007
 0091 00CE                 BSS  8
 0092               *
 0093 00D6 0C00     BINASK DATA >0C00                 BINARY TO HEX ASCII
 0094 00D8 0000     W1     DATA 0
 0095 00DA 0000     W2     DATA 0
 0096               *
```

```
0097 00DC 0000  OUTP1  DATA 0,>0B07,0,BUF,0   WRITE ASCII REC. ON LUNO 7
     00DE 0B07
     00E0 0000
     00E2 0034'
     00E4 0000
0098 00E6 0000  NCHAR1 DATA 0                 CHARACTER COUNT
0099            *
0100 00E8 0000  WEOF1  DATA 0,>0D07           WRITE EOF ON LUNO 7
     00EA 0D07
0101 00EC              BSS  8
0102                   END
```

```
 ' BINASK  00D6    ' BUF     0034    ' CS1IC   00CA    ' END1    00BE
 D EOFCS1  0010    ' NCHAR1  00E6    D OUCS1E  0004    D OUCS1I  0000
 D OUCS1L  000C    D OUCS1R  0008    ' OUTP1   00DC    ' S1      0096
 ' S2      00A2    ' START1  0084    ' W1      00D8    ' W2      00DA
 ' WEOF1   00E8    ' WSP     0014
```

0000 ERRORS

APPENDIX G

PSMAIN: FORTRAN LISTING

```
C   MAIN PROGRAM (SIMULATION)
C
       COMMON/OUT/ FLTP(8)
       COMMON/XXX/ MAXCNT,NERR,INDAT,OUTDAT
       COMMON/LOGL/ CHC,CHCHAN,EST,MLE
       COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E2P,F(5,13),
     1              GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(88),SGANS,
     2              X(5,10),XS(2,10),XY(2,3),Y(3),YP(3)
       COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GSQL(3),
     1              TJ(5),TIME,TL(5)
       COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP(5,5),SZP2(5,5),
     1              THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,ZP1MAX,ZP1MIN,
     2              ZP2,ZP2MAX,ZP2MIN,Z1MIN
       LOGICAL CHC,CHCHAN,EST,MLE
       INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X,XS,XY,Y,YP
       INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,TJ,TIME,TL
       INTEGER OUTDAT,CYCTIM
       INTEGER I1(3),I2(3),I3(3),I4(3),I5(3),I6(3)
       INTEGER I7(3),I8(6),I9(3),I10(3),I11(3),I12(3)
       DATA I1/6HNPARAM/,I2/6H NCHAN/,I3/6H   CHC/,I4/6H   EST/
       DATA I5/6H   MLE/,I6/6H NLOOP/,I7/6HCYCTIM/,I8/12H INDEVOUTDEV/
       DATA I9/6HMAXTIM/,I10/6HNERROR/,I11/6HIC CS1/,I12/6HNWORDS/
C
C   INITIAL CONDITION DATA FROM CS1
C
    10 CALL PRASRI(NP,0,I1)
       CALL PRASRI(NP,0,I1)
       CALL INASRI(NP,1,I11)
       CALL INCS1I(C1HP)
       CALL INCS1E(ANS)
       CALL INCS1R(DZP)
C
C   INPUT DATA FROM ASR
C
       CALL INASRI(NP,1,I1)
       CALL INASRI(NC,1,I2)
       CALL INASRL(CHC,1,I3)
       CALL INASRL(EST,1,I4)
       CALL INASRL(MLE,1,I5)
       CALL INASRI(NWORDS,1,I12)
       CALL INASRI(N,1,I6)
       CALL INASRI(CYCTIM,1,I7)
       CALL INASRI(INDAT,2,I8)
       CALL PRASRI(NP,0,I1)
       NPULSE = CYCTIM*3/25
       CALL TIMEON(NPULSE)
C
       DO 200 I=1,N
C
C   READ INPUT DATA
C
       IF(INDAT.EQ.8) CALL INCS2I(Y)
       IF(INDAT.EQ.9300) CALL R9300I(Y,3)
```

```
      CALL PCMLE
      IF(MODE.NE.1) GO TO 100
      FLTP(1) = ZP1
      FLTP(2) = ZP2
      FLTP(3) = JS
      FLTP(4) = TJ(1)
      FLTP(5) = TJ(2)
      FLTP(6) = TJ(3)
      FLTP(7) = TJ(4)
      FLTP(8) = TJ(5)
C
C  WRITE OUTPUT DATA
C
      IF(OUTDAT.EQ.7) CALL OUCS1R(FLTP,NWORDS)
      IF(OUTDAT.EQ.9300) CALL W9300R(FLTP,NWORDS)
  100 CALL WAIT
C
  200 CONTINUE
C
      CALL TIMEOF(MAXCNT,NERR)
      IF(OUTDAT.EQ.7) CALL EOFCS1
      MAXTIM = MAXCNT*25/3
      CALL PRASRI(MAXTIM,1,I9)
      CALL PRASRI(NERR,1,I10)
      GO TO 10
      END
```

```
COMMON BLOCK/OUT    / ALLOCATION  0020 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 FLTP   REAL           32 ARRAY

COMMON BLOCK/XXX    / ALLOCATION  0008 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 MAXCNT INTEGER*2       2 SCALAR    0002 NERR   INTEGER*2       2 SCALAR
0004 INDAT  INTEGER*2       2 SCALAR    0006 OUTDAT INTEGER*2       2 SCALAR

COMMON BLOCK/LOGL   / ALLOCATION  0008 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 CHC    LOGICAL         2 SCALAR    0002 CHCHAN LOGICAL         2 SCALAR
0004 EST    LOGICAL         2 SCALAR    0006 MLE    LOGICAL         2 SCALAR

COMMON BLOCK/INT2   / ALLOCATION  046A BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 C1HP   INTEGER*2       2 SCALAR    0002 C2HP   INTEGER*2       2 SCALAR
0004 D      INTEGER*2     160 ARRAY     00A4 DS     INTEGER*2     320 ARRAY
01E4 E1P    INTEGER*2       2 SCALAR    01E6 E2P    INTEGER*2       2 SCALAR
01E8 F      INTEGER*2     130 ARRAY     026A GK     INTEGER*2     160 ARRAY
030A JS     INTEGER*2       2 SCALAR    030C JSTEMP INTEGER*2       2 SCALAR
030E MODE   INTEGER*2       2 SCALAR    0310 NC     INTEGER*2       2 SCALAR
0312 NP     INTEGER*2       2 SCALAR    0314 Q      INTEGER*2     176 ARRAY
03C4 SGANS  INTEGER*2       2 SCALAR    03C6 X      INTEGER*2     100 ARRAY
042A XS     INTEGER*2      40 ARRAY     0452 XY     INTEGER*2      12 ARRAY
045E Y      INTEGER*2       6 ARRAY     0464 YP     INTEGER*2       6 ARRAY

COMMON BLOCK/INT4   / ALLOCATION  0060 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 ANS    INTEGER*4       4 SCALAR    0004 ANSI   INTEGER*4       4 SCALAR
0008 DT     INTEGER*4       4 SCALAR    000C GE     INTEGER*4       8 ARRAY
0014 GL     INTEGER*4       8 ARRAY     001C GSQE   INTEGER*4      12 ARRAY
0028 GSQL   INTEGER*4      12 ARRAY     0034 TJ     INTEGER*4      20 ARRAY
0048 TIME   INTEGER*4       4 SCALAR    004C TL     INTEGER*4      20 ARRAY

COMMON BLOCK/REAL   / ALLOCATION  0144 BYTES

LOCN NAME   MODE        BYTES TYPE      LOCN NAME   MODE        BYTES TYPE

0000 DZP    REAL            8 ARRAY     0008 GSQLO  REAL            8 ARRAY
0010 RTJC   REAL            4 SCALAR    0014 RTJS   REAL            4 SCALAR
0018 RTJZ   REAL            4 SCALAR    001C SZP    REAL          100 ARRAY
0080 SZP2   REAL          100 ARRAY     00E4 THRTJC REAL            4 SCALAR
00E8 THRTJZ REAL            4 SCALAR    00EC ZP     REAL           40 ARRAY
```

```
0114 ZPS    REAL           20 ARRAY    0128 ZP1    REAL           4 SCALAR
012C ZP1MAX REAL            4 SCALAR   0130 ZP1MIN REAL           4 SCALAR
0134 ZP2    REAL            4 SCALAR   0138 ZP2MAX REAL           4 SCALAR
013C ZP2MIN REAL            4 SCALAR   0140 Z1MIN  REAL           4 SCALAR


ARRAY ALLOCATION

LOCN NAME   MODE       BYTES TYPE      LOCN NAME   MODE       BYTES TYPE

0030 I1     INTEGER*2      6 ARRAY     0036 I2     INTEGER*2      6 ARRAY
003C I3     INTEGER*2      6 ARRAY     0042 I4     INTEGER*2      6 ARRAY
0048 I5     INTEGER*2      6 ARRAY     004E I6     INTEGER*2      6 ARRAY
0054 I7     INTEGER*2      6 ARRAY     005A I8     INTEGER*2     12 ARRAY
0066 I9     INTEGER*2      6 ARRAY     006C I10    INTEGER*2      6 ARRAY
0072 I11    INTEGER*2      6 ARRAY     0078 I12    INTEGER*2      6 ARRAY

SCALAR ALLOCATION

LOCN NAME   MODE       BYTES TYPE      LOCN NAME   MODE       BYTES TYPE

007E NWORDS INTEGER*2      2 SCALAR    0080 N      INTEGER*2      2 SCALAR
0082 CYCTIM INTEGER*2      2 SCALAR    0084 NPULSE INTEGER*2      2 SCALAR
0086 I      INTEGER*2      2 SCALAR    0088 MAXTIM INTEGER*2      2 SCALAR
```

SUBPROGRAMS CALLED

NAME	TYPE	ARGS	NAME	TYPE	ARGS	NAME	TYPE	ARGS
PRASRI	REAL	3	INASRI	INTEGER*2	3	INCS1I	INTEGER*2	1
INCS1E	INTEGER*2	1	INCS1R	INTEGER*2	1	INASRL	INTEGER*2	3
TIMEON	REAL	1	INCS2I	INTEGER*2	1	R9300I	REAL	2
PCMLE	REAL	0	OUCS1R	REAL	2	W9300R	REAL	2
WAIT	REAL	0	TIMEOF	REAL	2	EOFCS1	REAL	0
F$RREL	RUNTIME		F$REVP	RUNTIME		F$XPRE	RUNTIME	
F$R1DV	RUNTIME		F$RITP	RUNTIME		F$REL	RUNTIME	

STATEMENT LABELS

LOCN	LABEL	USE	LOCN	LABEL	USE	LOCN	LABEL	USE
0010	10		01EE	200	DO END	01E8	100	
00F4	M3		0106	M4		01D6	M5	
01E8	M6		00E4	M7		00F4	M8	
0106	M9		01E8	M10		01D6	M11	
01E8	M12		0212	M13		0212	M14	

STATEMENT LOCATIONS

LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN	LINE	LOCN
3	0010	4	0010	5	0010	6	0010	9	0010	11	0010
14	0010	15	0010	16	0010	17	0010	18	0010	19	0010
20	0010	21	0010	22	0010	26	0010	27	001C	28	0028
29	0034	30	003C	31	0044	35	004C	36	0058	37	0064
38	0070	39	007C	40	0088	41	0094	42	00A0	43	00AC
44	00B8	45	00C4	46	00D6	48	00DE	52	00E4	53	00F4
54	0106	55	010C	56	0114	57	0120	58	012C	59	0138
60	014A	61	0168	62	0186	63	01A4	67	01C2	68	01D6
69	01E8	71	01EE	73	01FA	74	0204	75	0212	76	0224
77	0230	78	023C	79	0240						

ENTRY=0006
PROGRAM SIZE=0254 BYTES
DATA SIZE=0092 BYTES
COMPILATION COMPLETE
0 WARNINGS
0 ERRORS

APPENDIX H

PSMAIN: TI990 ASSEMBLY LISTING

```
                     IDT   '$MAIN '
                     DEF   $MAIN
                     REF   PRASRI
                     REF   INASRI
                     REF   INCS1I
                     REF   INCS1E
                     REF   INCS1R
                     REF   INASRL
                     REF   TIMEON
                     REF   INCS2I
                     REF   R9300I
                     REF   PCMLE
                     REF   OUCS1R
                     REF   W9300R
                     REF   WAIT
                     REF   TIMEOF
                     REF   EOFCS1
0000                 CSEG  'OUT    '
0000         FLTP    BSS     32
0020                 CEND
0000                 CSEG  'XXX    '
0000         MAXCNT  BSS      2
0002         NERR    BSS      2
0004         INDAT   BSS      2
0006         OUTDAT  BSS      2
0008                 CEND
0000                 CSEG  'LOGL   '
0000         CHC     BSS      2
0002         CHCHAN  BSS      2
0004         EST     BSS      2
0006         MLE     BSS      2
0008                 CEND
0000                 CSEG  'INT2   '
0000         C1HP    BSS      2
0002         C2HP    BSS      2
0004         D       BSS    160
00A4         DS      BSS    320
01E4         E1P     BSS      2
01E6         E2P     BSS      2
01E8         F       BSS    130
026A         GK      BSS    160
030A         JS      BSS      2
030C         JSTEMP  BSS      2
030E         MODE    BSS      2
0310         NC      BSS      2
0312         NP      BSS      2
0314         Q       BSS    176
03C4         SGANS   BSS      2
03C6         X       BSS    100
042A         XS      BSS     40
0452         XY      BSS     12
045E         Y       BSS      6
0464         YP      BSS      6
046A                 CEND
```

```
0000              CSEG  'INT4    '
0000      ANS     BSS        4
0004      ANSI    BSS        4
0008      DT      BSS        4
000C      GE      BSS        8
0014      GL      BSS        8
001C      GSQE    BSS       12
0028      GSQL    BSS       12
0034      TJ      BSS       20
0048      TIME    BSS        4
004C      TL      BSS       20
0060              CEND
0000              CSEG  'REAL    '
0000      DZP     BSS        8
0008      GSQLO   BSS        8
0010      RTJC    BSS        4
0014      RTJS    BSS        4
0018      RTJZ    BSS        4
001C      SZP     BSS      100
0080      SZP2    BSS      100
00E4      THRTJC  BSS        4
00E8      THRTJZ  BSS        4
00EC      ZP      BSS       40
0114      ZPS     BSS       20
0128      ZP1     BSS        4
012C      ZP1MAX  BSS        4
0130      ZP1MIN  BSS        4
0134      ZP2     BSS        4
0138      ZP2MAX  BSS        4
013C      ZP2MIN  BSS        4
0140      Z1MIN   BSS        4
0144              CEND
0000              DSEG
0000      $DATA   BSS       48
0030      I1      BSS        6
0036      I2      BSS        6
003C      I3      BSS        6
0042      I4      BSS        6
0048      I5      BSS        6
004E      I6      BSS        6
0054      I7      BSS        6
005A      I8      BSS       12
0066      I9      BSS        6
006C      I10     BSS        6
0072      I11     BSS        6
0078      I12     BSS        6
007E      NWORDS  BSS        2
0080      N       BSS        2
0082      CYCTIM  BSS        2
0084      NPULSE  BSS        2
0086      I       BSS        2
0088      MAXTIM  BSS        2
008A              DEND
                  REF   F$RREL
```

```
                             LOAD F$RREL
     0000         $MAIN      PSEG
     0000 0000"              DATA $DATA
     0002 0006'              DATA $MAIN+>0006
                             REF  F$REVP
     0004 0000               DATA F$REVP
     0006                    RORG >0006
                             REF  F$XPRE
     0006 06A0               BL   @F$XPRE
     0008 0000
     000A   24               TEXT '$MAIN '
                  * 0001 C  MAIN PROGRAM (SIMULATION)
                  * 0002 C
                  * 0003        COMMON/OUT/ FLTP(8)
                  * 0004        COMMON /XXX/ MAXCNT,NERR,INDAT,OUTDAT
                  * 0005        COMMON/LOGL/ CHC,CHCHAN,EST,MLE
                  * 0006        COMMON/INT2/ C1HP,C2HP,D(5,16),DS(5,2,16),E1P,E
                  * 0007       1             GK(5,2,8),JS,JSTEMP,MODE,NC,NP,Q(8
                  * 0008       2             X(5,10),XS(2,10),XY(2,3),Y(3),YP(3
                  * 0009        COMMON/INT4/ ANS,ANSI,DT,GE(2),GL(2),GSQE(3),GS
                  * 0010       1             TJ(5),TIME,TL(5)
                  * 0011        COMMON/REAL/ DZP(2),GSQLO(2),RTJC,RTJS,RTJZ,SZP
                  * 0012       1             THRTJC,THRTJZ,ZP(5,2),ZPS(5),ZP1,Z
                  * 0013       2             ZP2,ZP2MAX,ZP2MIN,Z1MIN
                  * 0014        LOGICAL CHC,CHCHAN,EST,MLE
                  * 0015        INTEGER C1HP,C2HP,D,DS,E1P,E2P,F,GK,Q,S,SGANS,X
                  * 0016        INTEGER*4 ANS,ANSI,DT,GE,GL,GSQE,GSQL,MD,MS,SD,
                  * 0017        INTEGER OUTDAT,CYCTIM
                  * 0018        INTEGER I1(3),I2(3),I3(3),I4(3),I5(3),I6(3)
                  * 0019        INTEGER I7(3),I8(6),I9(3),I10(3),I11(3),I12(3)
                  * 0020        DATA I1/6HNPARAM/,I2/6H NCHAN/,I3/6H   CHC/,I4/
     008A                    DSEG
     0030                    RORG 48
     0030 4E50               DATA >4E50
     0032 4152               DATA >4152
     0034 414D               DATA >414D
     0036                    DEND
     0036                    DSEG
     0036                    RORG 54
     0036 204E               DATA >204E
     0038 4348               DATA >4348
     003A 414E               DATA >414E
     003C                    DEND
     003C                    DSEG
     003C                    RORG 60
     003C 2020               DATA >2020
     003E 2043               DATA >2043
     0040 4843               DATA >4843
     0042                    DEND
     0042                    DSEG
     0042                    RORG 66
     0042 2020               DATA >2020
     0044 2045               DATA >2045
     0046 5354               DATA >5354
```

```
0048                   DEND
              * 0021            DATA I5/6H   MLE/,I6/6H NLOOP/,I7/6HCYCTIM/,I8/
0048                   DSEG
0048                   RORG 72
0048 2020              DATA >2020
004A 204D              DATA >204D
004C 4C45              DATA >4C45
004E                   DEND
004E                   DSEG
004E                   RORG 78
004E 204E              DATA >204E
0050 4C4F              DATA >4C4F
0052 4F50              DATA >4F50
0054                   DEND
0054                   DSEG
0054                   RORG 84
0054 4359              DATA >4359
0056 4354              DATA >4354
0058 494D              DATA >494D
005A                   DEND
005A                   DSEG
005A                   RORG 90
005A 2049              DATA >2049
005C 4E44              DATA >4E44
005E 4556              DATA >4556
0060 4F55              DATA >4F55
0062 5444              DATA >5444
0064 4556              DATA >4556
0066                   DEND
              * 0022            DATA I9/6HMAXTIM/,I10/6HNERROR/,I11/6HIC CS1/,I
0066                   DSEG
0066                   RORG 102
0066 4D41              DATA >4D41
0068 5854              DATA >5854
006A 494D              DATA >494D
006C                   DEND
006C                   DSEG
006C                   RORG 108
006C 4E45              DATA >4E45
006E 5252              DATA >5252
0070 4F52              DATA >4F52
0072                   DEND
0072                   DSEG
0072                   RORG 114
0072 4943              DATA >4943
0074 2043              DATA >2043
0076 5331              DATA >5331
0078                   DEND
0078                   DSEG
0078                   RORG 120
0078 4E57              DATA >4E57
007A 4F52              DATA >4F52
007C 4453              DATA >4453
007E                   DEND
```

```
                    * 0023 C
                    * 0024 C  INITIAL CONDITION DATA FROM CS1
                    * 0025 C
                    * 0026    10 CALL PRASRI(NP,0,I1)
0010                       RORG >0010
          0010' $10        EQU  $
0010 0420                  BLWP @PRASRI
0012 0000                       .
0014 0003                  DATA 3
0016 0312+                 DATA NP
0018 0244'                 DATA I$3
001A 0030"                 DATA I1
                    * 0027       CALL PRASRI(NP,0,I1)
001C 0420                  BLWP @PRASRI
001E 0012'
0020 0003                  DATA 3
0022 0312+                 DATA NP
0024 0244'                 DATA I$3
0026 0030"                 DATA I1
                    * 0028       CALL INASRI(NP,1,I11)
0028 0420                  BLWP @INASRI
002A 0000
002C 0003                  DATA 3
002E 0312+                 DATA NP
0030 0246'                 DATA I$4
0032 0072"                 DATA I11
                    * 0029       CALL INCS1I(C1HP)
0034 0420                  BLWP @INCS1I
0036 0000
0038 0001                  DATA 1
003A 0000+                 DATA C1HP
                    * 0030       CALL INCS1E(ANS)
003C 0420                  BLWP @INCS1E
003E 0000
0040 0001                  DATA 1
0042 0000+                 DATA ANS
                    * 0031       CALL INCS1R(DZP)
0044 0420                  BLWP @INCS1R
0046 0000
0048 0001                  DATA 1
004A 0000+                 DATA DZP
                    * 0032 C
                    * 0033 C  INPUT DATA FROM ASR
                    * 0034 C
                    * 0035       CALL INASRI(NP,1,I1)
004C 0420                  BLWP @INASRI
004E 002A'
0050 0003                  DATA 3
0052 0312+                 DATA NP
0054 0246'                 DATA I$4
0056 0030"                 DATA I1
                    * 0036       CALL INASRI(NC,1,I2)
0058 0420                  BLWP @INASRI
005A 004E'
```

```
005C 0003           DATA 3
005E 0310+          DATA NC
0060 0246'          DATA I$4
0062 0036"          DATA I2
           * 0037       CALL INASRL(CHC,1,I3)
0064 0420           BLWP @INASRL
0066 0000
0068 0003           DATA 3.
006A 0000+          DATA CHC
006C 0246'          DATA I$4
006E 003C"          DATA I3
           * 0038       CALL INASRL(EST,1,I4)
0070 0420           BLWP @INASRL
0072 0066'
0074 0003           DATA 3
0076 0004+          DATA EST
0078 0246'          DATA I$4
007A 0042"          DATA I4
           * 0039       CALL INASRL(MLE,1,I5)
007C 0420           BLWP @INASRL
007E 0072'
0080 0003           DATA 3
0082 0006+          DATA MLE
0084 0246'          DATA I$4
0086 0048"          DATA I5
           * 0040       CALL INASRI(NWORDS,1,I12)
0088 0420           BLWP @INASRI
008A 005A'
008C 0003           DATA 3
008E 007E"          DATA NWORDS
0090 0246'          DATA I$4
0092 0078"          DATA I12
           * 0041       CALL INASRI(N,1,I6)
0094 0420           BLWP @INASRI
0096 008A'
0098 0003           DATA 3
009A 0080"          DATA N
009C 0246'          DATA I$4
009E 004E"          DATA I6
           * 0042       CALL INASRI(CYCTIM,1,I7)
00A0 0420           BLWP @INASRI
00A2 0096'
00A4 0003           DATA 3
00A6 0082"          DATA CYCTIM
00A8 0246'          DATA I$4
00AA 0054"          DATA I7
           * 0043       CALL INASRI(INDAT,2,I8)
00AC 0420           BLWP @INASRI
00AE 00A2'
00B0 0003           DATA 3
00B2 0004+          DATA INDAT
00B4 0248'          DATA I$2
00B6 005A"          DATA I8
           * 0044       CALL PRASRI(NP,0,I1)
```

```
00B8 0420          BLWP @PRASRI
00BA 001E'
00BC 0003          DATA 3
00BE 0312+         DATA NP
00C0 0244'         DATA I$3
00C2 0030"         DATA I1
          * 0045        NPULSE = CYCTIM*3/25
00C4 C020          MOV  @CYCTIM,0
00C6 0082"
00C8 3820          MPY  @I$5,0
00CA 024A'
                   REF  F$R1DV
00CC 06A0          BL   @F$R1DV
00CE 0000
00D0 024C'         DATA I$6
00D2 C801          MOV  1,@NPULSE
00D4 0084"
          * 0046        CALL TIMEON(NPULSE)
00D6 0420          BLWP @TIMEON
00D8 0000
00DA 0001          DATA 1
00DC 0084"         DATA NPULSE
          * 0047 C
          * 0048        DO 200 I=1,N
00DE C820          MOV  @I$4,@I
00E0 0246'
00E2 0086"
     00E4' M$7     EQU  $
          * 0049 C
          * 0050 C  READ INPUT DATA
          * 0051 C
          * 0052        IF(INDAT.EQ.8) CALL INCS2I(Y)
00E4 8820          C    @INDAT,@I$7
00E6 0004+
00E8 024E'
00EA 1604          JNE  M$9
00EC 0420          BLWP @INCS2I
00EE 0000
00F0 0001          DATA 1
00F2 045E+         DATA Y
     00F4' M$3     EQU  $
     00F4' M$8     EQU  $
     00F4' M$9     EQU  M$8
          * 0053        IF(INDAT.EQ.9300) CALL R9300I(Y,3)
00F4 8820          C    @INDAT,@I$8
00F6 0004+
00F8 0250'
00FA 1605          JNE  M$11
00FC 0420          BLWP @R9300I
00FE 0000
0100 0002          DATA 2
0102 045E+         DATA Y
0104 024A'         DATA I$5
     0106' M$4     EQU  $
```

```
0349      0106' M$10    EQU  $
0350      0106' M$11    EQU  M$10
0351            * 0054       CALL PCMLE
0352 0106 0420          BLWP @PCMLE
     0108 0000
0353 010A 0000          DATA 0
0354            * 0055       IF(MODE.NE.1) GO TO 100
0355 010C 8820          C    @MODE,@I$4
     010E 030E+
     0110 0246'
0356 0112 166A          JNE  M$13
0357            * 0056       FLTP(1) = ZP1
0358 0114 C820          MOV  @ZP1,@FLTP
     0116 0128+
     0118 0000+
0359 011A C820          MOV  @ZP1+2,@FLTP+2
     011C 012A+
     011E 0002+
0360            * 0057       FLTP(2) = ZP2
0361 0120 C820          MOV  @ZP2,@FLTP+4
     0122 0134+
     0124 0004+
0362 0126 C820          MOV  @ZP2+2,@FLTP+4+2
     0128 0136+
     012A 0006+
0363            * 0058       FLTP(3) = JS
0364                    REF  F$RITP
0365 012C 0420          BLWP @F$RITP
     012E 0000
0366 0130 0CA0          CIR  @JS
     0132 030A+
0367 0134 0DE0          STR  @FLTP+8
     0136 0008+
0368            * 0059       FLTP(4) = TJ(1)
0369                    REF  F$REL
0370 0138 0C0E          XIT
0371 013A 0420          BLWP @F$REL
     013C 0000
0372 013E 0034+         DATA TJ
0373 0140 0420          BLWP @F$RITP
     0142 012E'
0374 0144 0C06          CER
0375 0146 0DE0          STR  @FLTP+12
     0148 000C+
0376            * 0060       FLTP(5) = TJ(2)
0377 014A 0C0E          XIT
0378 014C 0202          LI   2,4
     014E 0004
0379 0150 0222          AI   2,TJ
     0152 0034+
0380 0154 C802          MOV  2,@T$+0
     0156 007E"
0381 0158 0420          BLWP @F$REL
     015A 013C'
```

```
0382 015C 007F"         DATA T$+0+1
0383 015E 0420          BLWP @F$RITP
     0160 0142'
0384 0162 0C06          CER
0385 0164 0DE0          STR  @FLTP+16
     0166 0010+
0386              * 0061     FLTP(6) = TJ(3)
0387 0168 0C0E          XIT
0388 016A 0202          LI   2,8
     016C 0008
0389 016E 0222          AI   2,TJ
     0170 0034+
0390 0172 C802          MOV  2,@T$+2
     0174 0080"
0391 0176 0420          BLWP @F$REL
     0178 015A'
0392 017A 0081"         DATA T$+2+1
0393 017C 0420          BLWP @F$RITP
     017E 0160'
0394 0180 0C06          CER
0395 0182 0DE0          STR  @FLTP+20
     0184 0014+
0396              * 0062     FLTP(7) = TJ(4)
0397 0186 0C0E          XIT
0398 0188 0202          LI   2,12
     018A 000C
0399 018C 0222          AI   2,TJ
     018E 0034+
0400 0190 C802          MOV  2,@T$+4
     0192 0082"
0401 0194 0420          BLWP @F$REL
     0196 0178'
0402 0198 0083"         DATA T$+4+1
0403 019A 0420          BLWP @F$RITP
     019C 017E'
0404 019E 0C06          CER
0405 01A0 0DE0          STR  @FLTP+24
     01A2 0018+
0406              * 0063     FLTP(8) = TJ(5)
0407 01A4 0C0E          XIT
0408 01A6 0202          LI   2,16
     01A8 0010
0409 01AA 0222          AI   2,TJ
     01AC 0034+
0410 01AE C802          MOV  2,@T$+6
     01B0 0084"
0411 01B2 0420          BLWP @F$REL
     01B4 0196'
0412 01B6 0085"         DATA T$+6+1
0413 01B8 0420          BLWP @F$RITP
     01BA 019C'
0414 01BC 0C06          CER
0415 01BE 0DE0          STR  @FLTP+28
     01C0 001C+
```

```
                * 0064 C
                * 0065 C  WRITE OUTPUT DATA
                * 0066 C
                * 0067        IF(OUTDAT.EQ.7) CALL OUCS1R(FLTP,NWORDS)
01C2 0C0E              XIT
01C4 8820              C     @OUTDAT,@I$9
01C6 0006+
01C8 0252'
01CA 1605              JNE  M$15
01CC 0420              BLWP @OUCS1R
01CE 0000
01D0 0002              DATA 2
01D2 0000+             DATA FLTP
01D4 007E"             DATA NWORDS
     01D6' M$5         EQU  $
     01D6' M$14        EQU  $
     01D6' M$15        EQU  M$14
                * 0068        IF(OUTDAT.EQ.9300) CALL W9300R(FLTP,NWORDS)
01D6 8820              C     @OUTDAT,@I$8
01D8 0006+
01DA 0250'
01DC 1605              JNE  M$17
01DE 0420              BLWP @W9300R
01E0 0000
01E2 0002              DATA 2
01E4 0000+             DATA FLTP
01E6 007E"             DATA NWORDS
     01E8' M$6         EQU  $
     01E8' M$16        EQU  $
     01E8' M$17        EQU  M$16
                * 0069    100 CALL WAIT
     01E8' $100        EQU  $
     01E8' M$12        EQU  $
     01E8' M$13        EQU  M$12
01E8 0420              BLWP @WAIT
01EA 0000
01EC 0000              DATA 0
                * 0070 C
                * 0071    200 CONTINUE
     01EE' $200        EQU  $
01EE 05A0              INC  @I
01F0 0086"
01F2 8820              C     @I,@N
01F4 0086"
01F6 0080"
01F8 1223              JLE  M$19
                * 0072 C
                * 0073        CALL TIMEOF(MAXCNT,NERR)
01FA 0420              BLWP @TIMEOF
01FC 0000
01FE 0002              DATA 2
0200 0000+             DATA MAXCNT
0202 0002+             DATA NERR
                * 0074        IF(OUTDAT.EQ.7) CALL EOFCS1
```

```
0459 0204 8820           C     @OUTDAT,@I$9
     0206 0006+
     0208 0252'
0460 020A 1603           JNE   M$21
0461 020C 0420           BLWP  @EOFCS1
     020E 0000
0462 0210 0000           DATA  0
0463      0212' M$18     EQU   $
0464      0212' M$20     EQU   $
0465      0212' M$21     EQU   M$20
0466            * 0075          MAXTIM = MAXCNT*25/3
0467 0212 C020           MOV   @MAXCNT,0
     0214 0000+
0468 0216 3820           MPY   @I$6,0
     0218 024C'
0469 021A 06A0           BL    @F$R1DV
     021C 00CE'
0470 021E 024A'          DATA  I$5
0471 0220 C801           MOV   1,@MAXTIM
     0222 0088"
0472            * 0076          CALL PRASRI(MAXTIM,1,I9)
0473 0224 0420           BLWP  @PRASRI
     0226 00BA'
0474 0228 0003           DATA  3
0475 022A 0088"          DATA  MAXTIM
0476 022C 0246'          DATA  I$4
0477 022E 0066"          DATA  I9
0478            * 0077          CALL PRASRI(NERR,1,I10)
0479 0230 0420           BLWP  @PRASRI
     0232 0226'
0480 0234 0003           DATA  3
0481 0236 0002+          DATA  NERR
0482 0238 0246'          DATA  I$4
0483 023A 006C"          DATA  I10
0484            * 0078          GO TO 10
0485      023C' M$22     EQU   $
0486 023C 0460           B     @$10
     023E 0010'
0487            * 0079          END
0488      0240' M$19     EQU   $
0489 0240 0460           B     @M$7
     0242 00E4'
0490 0244 0000  I$3      DATA  0
0491 0246 0001  I$4      DATA  1
0492 0248 0002  I$2      DATA  2
0493 024A 0003  I$5      DATA  3
0494 024C 0019  I$6      DATA  25
0495 024E 0008  I$7      DATA  8
0496 0250 2454  I$8      DATA  9300
0497 0252 0007  I$9      DATA  7
0498 007E                DSEG
0499 007E       T$       BSS   8
0500 0086                DEND
0501                     END   $MAIN
```

'	$10	0010	'	$100	01E8	'	$200	01EE	"	$DATA	0000
D	$MAIN	0000	+	ANS	0000	+	ANSI	0004	+	C1HP	0000
+	C2HP	0002	+	CHC	0000	+	CHCHAN	0002	"	CYCTIM	0082
+	D	0004	+	DS	00A4	+	DT	0008	+	DZP	0000
+	E1P	01E4	+	E2P	01E6	E	EOFCS1	020E	+	EST	0004
+	F	01E8	E	F$R1DV	021C	E	F$REL	01B4	E	F$REVP	0004
E	F$RITP	01BA	E	F$RREL	0000	E	F$XPRE	0008	+	FLTP	0000
+	GE	000C	+	GK	026A	+	GL	0014	+	GSQE	001C
+	GSQL	0028	+	GSQLO	0008	"	I	0086	'	I$2	0248
'	I$3	0244	'	I$4	0246	'	I$5	024A	'	I$6	024C
'	I$7	024E	'	I$8	0250	'	I$9	0252	"	I1	0030
"	I10	006C	"	I11	0072	"	I12	0078	"	I2	0036
"	I3	003C	"	I4	0042	"	I5	0048	"	I6	004E
"	I7	0054	"	I8	005A	"	I9	0066	E	INASRI	00AE
E	INASRL	007E	E	INCS1E	003E	E	INCS1I	0036	E	INCS1R	0046
E	INCS2I	00EE	+	INDAT	0004	+	JS	030A	+	JSTEMP	030C
'	M$10	0106	'	M$11	0106	'	M$12	01E8	'	M$13	01E8
'	M$14	01D6	'	M$15	01D6	'	M$16	01E8	'	M$17	01E8
'	M$18	0212	'	M$19	0240	'	M$20	0212	'	M$21	0212
'	M$22	023C	'	M$3	00F4	'	M$4	0106	'	M$5	01D6
'	M$6	01E8	'	M$7	00E4	'	M$8	00F4	'	M$9	00F4
+	MAXCNT	0000	"	MAXTIM	0088	+	MLE	0006	+	MODE	030E
"	N	0080	+	NC	0310	+	NERR	0002	+	NP	0312
"	NPULSE	0084	"	NWORDS	007E	E	OUCS1R	01CE	+	OUTDAT	0006
E	PCMLE	0108	E	PRASRI	0232	+	Q	0314	E	R9300I	00FE
+	RTJC	0010	+	RTJS	0014	+	RTJZ	0018	+	SGANS	03C4
+	SZP	001C	+	SZP2	0080	"	T$	007E	+	THRTJC	00E4
+	THRTJZ	00E8	+	TIME	0048	E	TIMEOF	01FC	E	TIMEON	00D8
+	TJ	0034	+	TL	004C	E	W9300R	01E0	E	WAIT	01EA
+	X	03C6	+	XS	042A	+	XY	0452	+	Y	045E
+	YP	0464	+	Z1MIN	0140	+	ZP	00EC	+	ZP1	0128
+	ZP1MAX	012C	+	ZP1MIN	0130	+	ZP2	0134	+	ZP2MAX	0138
+	ZP2MIN	013C	+	ZPS	0114						

0000 ERRORS

APPENDIX I

PSPCMLE: LINK EDITOR LISTING

```
COMMAND LIST

NOSYMT
TASK FORT
INCL DSC2:MAIN/OBJ
INCL DSC2:PCMLE/OBJ
INCL DSC2:CLOCK5/OBJ
INCL DSC2:INPCAS/OBJ
INCL DSC2:IO9300/OBJ
INCL DSC2:IOASR/OBJ
INCL DSC2:MD/OBJ
INCL DSC2:MS/OBJ
INCL DSC2:OUTCAS/OBJ
INCL DSC2:S/OBJ
INCL DSC2:SD/OBJ
FIND :TXLOBJ/LIB
END
```

```
 TXSLNK        2.3.0  78.244       07/15/79  11:15:39                PAGE   2
LINK MAP

CONTROL FILE = DSC2:LINK/

LINKED OUTPUT FILE = CS1

LIST FILE = LP

OUTPUT FORMAT = ASCII

LIBRARIES

NO    ORGANIZATION    PATHNAME

1     SEQUENTIAL      :TXLOBJ/LIB
```

PHASE 0, FORT ORIGIN = 0000 LENGTH = 4FD8 ENTRY=0000

MODULE	NO	ORIGIN	LENGTH	TYPE	DATE	TIME	CREATOR
$MAIN	1	0000	0254	INCLUDE	07/13/79	08:45:28	TXFTN
$DATA	1	3DC0	0092				
PCMLE	2	0254	010E	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	2	3E52	0032				
FILT	3	0362	0770	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	3	3E84	0156				
SENS	4	0AD2	0B0C	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	4	3FDA	0102				
ACUM	5	15DE	0524	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	5	40DC	00E4				
FH	6	1B02	010E	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	6	41C0	0054				
CYC1	7	1C10	012E	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	7	4214	0052				
CYC2	8	1D3E	01B2	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	8	4266	0050				
CYC3	9	1EF0	01FC	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	9	42B6	004E				
CYC4	10	20EC	0118	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	10	4304	005E				
CYC5	11	2204	00B0	INCLUDE	01/00/00	00:15:06	TXFTN
$DATA	11	4362	0030				
CLOCK5	12	22B4	00B6	INCLUDE	01/01/00	00:38:57	XMIRA
INPCAS	13	236A	0134	INCLUDE	01/01/00	00:20:40	XMIRA
IO9300	14	249E	009E	INCLUDE	01/01/00	00:37:00	XMIRA
IOASR	15	253C	01A2	INCLUDE	06/18/79	10:33:56	XMIRA
MD	16	26DE	004A	INCLUDE	07/10/79	09:36:23	XMIRA
$DATA	16	4392	0020				
MS	17	2728	008A	INCLUDE	07/13/79	11:55:33	XMIRA
$DATA	17	43B2	0042				
OUTCAS	18	27B2	00F4	INCLUDE	01/01/00	00:19:01	XMIRA
S	19	28A6	004C	INCLUDE	07/10/79	09:33:32	XMIRA
$DATA	19	43F4	0020				
SD	20	28F2	006A	INCLUDE	07/13/79	11:56:42	XMIRA
$DATA	20	4414	0042				
F$XPRE	21	295C	0768	SEARCH,1	10/26/78	01:40:40	SDSLNK
$DATA	21	4456	01D4				
F$REVP	22	30C4	000C	SEARCH,1	10/25/78	23:22:31	SDSLNK
DFTLRL	23	30D0	0072	SEARCH,1	10/26/78	00:30:54	SDSLNK
F$RCGO	24	3142	0032	SEARCH,1	10/25/78	23:36:35	SDSLNK
F$RGMY	25	3174	007E	SEARCH,1	10/25/78	23:41:34	SDSLNK
F$RAER	26	31F2	0041	SEARCH,1	10/25/78	23:35:42	SDSLNK
F$RITP	27	3234	0010	SEARCH,1	10/25/78	23:43:26	SDSLNK
F$FITP	28	3244	01D2	SEARCH,1	10/25/78	23:26:51	SDSLNK
F$RITE	29	3416	006E	SEARCH,1	10/25/78	23:42:40	SDSLNK
IBIT	30	3484	00B4	SEARCH,1	10/25/78	23:47:18	SDSLNK
F$XREL	31	3538	001A	SEARCH,1	10/25/78	23:46:03	SDSLNK
F$FLT	32	3552	00A4	SEARCH,1	10/25/78	23:33:30	SDSLNK
F$PASR	33	35F6	0360	SEARCH,1	10/25/78	23:30:08	SDSLNK
F$FIX	34	3956	00D0	SEARCH,1	10/25/78	23:32:54	SDSLNK
EXINT	35	3A26	0172	SEARCH,1	10/25/78	23:18:41	SDSLNK
F$RWSP	36	3B98	0000	SEARCH,1	10/25/78	23:22:56	SDSLNK
$DATA	36	462A	00FC				

MODULE	NO	ORIGIN	LENGTH	TYPE	DATE	TIME	CREATOR
INTGR	37	3B98	00DA	SEARCH,1	10/25/78	23:50:47	SDSLNK
F$ERRC	38	3C72	013C	SEARCH,1	10/26/78	00:33:07	SDSLNK
F$XERR	39	3DAE	000A	SEARCH,1	10/26/78	00:57:52	SDSLNK
F$XTBLTX	40	3DB8	0000	SEARCH,1	10/26/78	01:39:04	SDSLNK
$DATA	40	4726	01E0				
F$XFTLTX	41	3DB8	0005	SEARCH,1	10/26/78	01:37:12	SDSLNK
F$RBUF	42	3DBE	0000	SEARCH,1	10/26/78	00:38:50	SDSLNK
$DATA	42	4906	0094				
F$XRST	43	3DBE	0002	SEARCH,1	10/25/78	23:23:41	SDSLNK

COMMON	NO	ORIGIN	LENGTH
OUT	1	499A	0020
XXX	1	49BA	0008
LOGL	11	49C2	0008
INT2	11	49CA	046A
INT4	11	4E34	0060
REAL	11	4E94	0144

D E F I N I T I O N S

NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO
$MAIN	0000	1	*A$BBUF	0126*	21	*A$BFCB	011A*	21	*A$BPRB	0122*	21
*A$BTCA	011E*	21	*A$BWK1	4456	21	A$BWK2	462A	36	*A$EFCB	011C*	21
*A$EPRB	0124*	21	A$ERRC	014C*	21	A$ERRS	014E*	21	*A$ETCA	0120*	21
A$XFTL	0150*	21	ACUM	15DE	5	CYC1	1C10	7	CYC2	1D3E	8
CYC3	1EF0	9	CYC4	20EC	10	CYC5	2204	11	DFTLRL	3106	23
EOFCS1	27C2	18	*F$ASAD	0006*	21	F$ERRC	3C72	38	F$ERRS	3C78	38
*F$ERST	3CFC	38	*F$FCBE	000A*	21	F$FITP	3244	28	*F$FLAG	0005*	21
F$ILOG	45A0	21	*F$LSTA	0001*	21	*F$LUNO	0000*	21	*F$NAME	0008*	21
*F$PRB	0002*	21	*F$R0DV	3B98	37	*F$R10A	0014*	21	*F$R10B	013C*	21
F$R1DV	3BA0	37	*F$R2DV	3BAA	37	*F$R3DV	3BB4	37	*F$R4DV	3BBE	37
*F$R5DV	3BC8	37	*F$R6DV	3BD2	37	*F$R7DV	3BDC	37	*F$R8DV	3BE6	37
*F$R9DV	3BF0	37	*F$RADV	3BFA	37	F$RAER	31F2	26	F$RBUF	490A	42
F$RCGO	3142	24	F$REA	3A46	35	*F$RECB	3B2C	35	*F$REDV	3A7A	35
F$REL	3A26	35	*F$REMP	3A58	35	*F$RENG	3B48	35	F$RES	3A34	35
*F$RESQ	3B18	35	F$RET	3B5C	35	F$REVP	30C4	22	*F$RFTE	3430	29
F$RGMY	3174	25	*F$RIBC	34BE	30	*F$RIBS	34AC	30	*F$RIBT	34CE	30
F$RISH	3484	30	F$RITE	3416	29	F$RITP	3234	27	F$RLOG	0148*	21
*F$RLP2	014A*	21	*F$RPAU	29AA	21	*F$RPRE	2A5A	21	*F$RREL	3538	31
F$RSTO	29BC	21	*F$RTFG	447E	21	*F$RVFB	0028*	21	*F$RVP2	002A*	21
F$RWRK	4458	21	F$RWSP	462C	36	*F$STAT	0004*	21	F$XAR	3600	33
*F$XBCS	010A*	21	F$XBFS	0090*	42	*F$XBUT	2F4A	21	F$XCDE	3962	34
F$XCDI	395E	34	F$XCED	355E	32	F$XCER	355A	32	F$XCID	3556	32
F$XCIR	3552	32	*F$XCLS	2BC2	21	F$XCRE	395A	34	F$XCRI	3956	34
F$XDR	3800	33	F$XERR	3DAE	39	F$XFTL	3DB8	41	*F$XLIO	2F30	21
*F$XLOG	2E7C	21	F$XLR	3538	31	*F$XLWS	45AA	21	F$XMR	375C	33
F$XNGR	3544	31	F$XPRE	295C	21	*F$XPSE	2B5E	21	F$XRST	3DBE	43
*F$XSA1	0111*	21	*F$XSA2	0112*	21	F$XSR	35F8	33	*F$XSTC	010C*	21
*F$XSTL	010E*	21	*F$XSTP	2B66	21	F$XSTR	353E	31	*F$XSVC	0110*	21
F$XTBE	4906	40	F$XTBL	4726	40	*F$XTID	4571	21	*F$XVBF	0100*	21
*F$XVCC	00FB*	21	*F$XVCH	0104*	21	*F$XVCL	00FF*	21	*F$XVCO	00FC*	21
*F$XVFB	2C4C	21	*F$XVRC	0102*	21	*F$XVRO	00FE*	21	*F$XVST	00FD*	21

TXSLNK 2.3.0 78.244 07/15/79 11:15:39 PAGE 5

NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO
*F$XVSV	00FA*	21	*F$XVWS	4580	21	*F$XWS0	4652	36	*F$XWS1	4654	36
*F$XWS2	4656	36	*F$XWS3	4658	36	FH	1B02	6	FILT	0362	3
*G$XE01	2F78	21	*G$XE08	2F83	21	*G$XE09	2FA5	21	*G$XE10	2FD2	21
*G$XE11	2FED	21	*G$XE12	3008	21	*G$XE13	3020	21	G$XE14	305E	21
*G$XE15	3070	21	*INASRE	2540	15	INASRI	253C	15	INASRL	2548	15
*INASRR	2544	15	INCS1E	236E	13	INCS1I	236A	13	*INCS1L	2376	13
INCS1R	2372	13	*INCS2E	237E	13	INCS2I	237A	13	*INCS2L	2386	13
*INCS2R	2382	13	MD	26DE	16	MS	2728	17	*N$COLS	0106*	21
*N$LINS	0108*	21	*N$TID	0119*	21	*OUCS1E	27B6	18	*OUCS1I	27B2	18
*OUCS1L	27BE	18	OUCS1R	27BA	18	*P$ABUF	0006*	21	*P$CCNT	000A*	21
*P$ERR	0001*	21	*P$LACN	0016*	21	*P$LFIL	0011*	21	*P$LIBF	0010*	21
*P$LLRL	0012*	21	*P$LPRL	0014*	21	*P$LUN	0003*	21	*P$OP	0002*	21
*P$PFCB	0018*	21	*P$PRB	0000*	21	*P$PRBE	001A*	21	*P$REC1	000D*	21
*P$REC2	000E*	21	*P$RECL	0008*	21	*P$RES	000C*	21	*P$SFLG	0004*	21
*P$SVC0	0000*	21	*P$UFLG	0005*	21	*P$UTF1	0010*	21	*P$UTF2	0011*	21
PCMLE	0254	2	*PRASRE	2550	15	PRASRI	254C	15	*PRASRL	2558	15
*PRASRR	2554	15	R9300I	249E	14	S	28A6	19	*S$APRB	2D9C	21
SD	28F2	20	SENS	0AD2	4	TIMEOF	2326	12	TIMEON	22B4	12
*W9300I	24A2	14	W9300R	24A6	14	WAIT	230A	12			

**** LINKING COMPLETED